Sélection et composition flexible basée services abstraits pour une meilleure adaptation aux intentions des utilisateurs

Emma Fki

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Université Toulouse 1 Capitole (UT1 Capitole)

Présentée et soutenue par :

Emna FKI

le jeudi 17 Décembre 2015

Titre :

Sélection et composition flexible basée services abstraits pour une meilleure adaptation aux intentions des utilisateurs

---

École doctorale et discipline ou spécialité :

ED MITT : Domaine STIC : Réseaux, Télécoms, Systèmes et Architecture

Unité de recherche :

LAAS CNRS

Directeur/trice(s) de Thèse :

M. Said TAZI, Maitre de Conférences HDR, Université Toulouse 1 Capitole
M. Mohamed JMAIEL, Professeur, Université de Sfax

Jury :

Mme Ikram AMMOUS, Maitre de Conférences HDR, Université de Sfax (Rapporteur)
M. Djamal BEN SLIMANE, Professeur des Universités, Université Claude Bernard (Rapporteur)
M. Thierry VILLEMUR, Professeur des Universités, Université Toulouse 2 Jean Jaurès (Examinateur)
M. Khalil DRIRA, Directeur de recherche, LAAS CNRS (Examinateur)
M. Philippe ROOSE Maitre de Conférences HDR, Université de Pau et des Pays de l'Adour (Examinateur)

Emna Fki

SÉLECTION ET COMPOSITION FLEXIBLE BASÉE SERVICES ABSTRAITS POUR UNE MEILLEURE ADAPTATION AUX INTENTIONS DES UTILISATEURS

Directeurs de thèse :

Said TAZI, Mohamed JMAIEL

Résumé

La méthode de conception des architectures orientées services (SOA) est basée sur des standards et permet de créer une infrastructure informatique intégrée capable de répondre rapidement aux nouveaux besoins d'un utilisateur. Réellement, il n'est pas toujours facile de trouver des services correspondant aux requêtes des utilisateurs. Par conséquent, la composition des services satisfaisant la requête est un besoin grandissant de nos jours. La composition de services implique la capacité de sélectionner, de coordonner, d'interagir, et de faire interopérer des services existants. Elle constitue une tâche complexe. Cette complexité est due principalement au grand nombre de services disponibles et à leur hétérogénéité puisqu'ils sont créés par des organisations différentes. Cette complexité est renforcée quand il s'agit d'intégrer dynamiquement des services à la demande, et les composer automatiquement pour répondre à des exigences qui ne sont pas réalisées par les services existants.

En fait, une approche pour la composition de services doit offrir le potentiel de réaliser des applications flexibles et adaptables, en sélectionnant et en combinant les services de manière appropriée sur la base de la requête et du contexte de l'utilisateur. Dans cette perspective, différentes approches ont été développées pour la composition de services. Cependant, la plupart des processus de composition ont tendance à être statique et non flexible dans le sens où ils n'ont pas la capacité de s'adapter aux besoins des utilisateurs.

Dans cette thèse, nous proposons une approche de composition dans laquelle la génération du schéma de composition est effectuée en partie au moment de l'exécution en ayant recours aux services abstraits fournis au moment de la conception. L'utilisation des services abstraits permet une certaine flexibilité et adaptabilité sans avoir besoin de construire une composition de services à partir de zéro au moment de l'exécution. Le processus de composition que nous proposons se compose principalement de quatre étapes. La première étape prend une structuration des besoins de l'utilisateur matérialisée par un graphe d'intentions et l'enrichit pour expliciter les relations implicites. Le résultat de cette étape permet de générer un schéma de composition initial en construisant le flux de contrôle déduit du graphe des intentions enrichi, puis en sélectionnant les services abstraits adéquats. Le choix de ces services est basé sur le matching sémantique et le degré d'affinité sémantique entre les services abstraits. La troisième étape consiste à générer le schéma de composition final à l'aide d'un mécanisme de raffinement des services abstraits en utilisant des techniques de matching sémantique et en tenant compte du contexte de l'utilisateur. Enfin, le plan d'exécution est généré en tenant compte des contraintes non-fonctionnelles fournies dans la spécification des intentions.

Emna Fki

A FLEXIBLE COMPOSITION BASED ABSTRACT SERVICES FOR A BETTER ADAPTATION TO USERS' INTENTIONS

Supervisors :

Said TAZI, Mohamed JMAIEL

Abstract

The design approach of service oriented architectures (SOA) is based on standards which gives the possibility of creating an integrated IT infrastructure capable of rapidly responding to new user needs. Actually, it is not always easy to find services that meet user requests. Therefore, the service composition satisfying the user intention is a growing need. The composition of services implies the ability to select, coordinate, interact, and to interoperate existing services. The composition is considered as a complex task. This complexity is mainly due to the large number of available services and their heterogeneity as they are created by different organizations. This complexity is increased when services must be dynamically and automatically composed to meet requirements which are not satisfied by existing services. In fact, an approach for service composition must offer the potential to achieve flexible and adaptable applications, by selecting and combining services based of the request and the context of the user. In this perspective, different approaches have been developed for services composition. However, most of the existing composition approaches tend to be static and not flexible in the sense that they do not have the ability to adapt to user requirements.

In this thesis, we propose a composition approach in which the generation of the composition schema is performed at runtime through the use of abstract services provided at design time. The use of abstract services allows flexibility and adaptability without having to build a service composition from scratch at run time. The composition process that we propose consists mainly of four steps. The first step takes a structure of user requirements materialized by a graph of intentions and enriches this graph to explicit the implicit relationships. The enriched graph is used to generate an initial composition scheme by building the control flow and selecting the appropriate abstract services. The selection of these services is based on the semantic matching and the degree of semantic affinity between abstract services. The third step is to generate the final composition schema with a refinement mechanism of abstract services using semantic matching techniques and taking into account user context and constraints. Finally, the execution plan is generated driven by non-functional constraints provided in the intentions specification.

Remerciements

JE remercie tout d'abord mes directeurs de tèhse M. Said TAZI et M. Mohamed JMAIEL pour l'encadrement de mon travail et pour leurs conseils avisés et leurs remarques pertinentes afin d'améliorer la qualité de ma tèhse.

Je voudrais ensuite exprimer ma reconnaissance à M. Djamal BENSLIMANE et Mme Ikram AMMOUS, qui ont accepté de rapporter mon manuscrit. Je remercie également M. Khalil DRIRA, M. Thierry VILLEMUR et M. Philippe ROOSE pour être examinateurs de mon travail.

Mes remerciements vont aussi à tous les chercheurs, enseignants et membres du personnel du laboratoire ReDCAD à Sfax et du laboratoire LAAS à Toulouse pour leur amitié et leur aide pendant ces années de thèse.

J'adresse ma plus grande gratitude à mes collègues et amis pour les bons moments que nous avons vécus ensemble.

Table des matières

Introduction

LES dernières décennies ont été marquées par le développement rapide des systèmes d'information distribués, et la diffusion de l'accès à Internet. Cette évolution du monde informatique a entraîné le développement de nouveaux paradigmes d'interaction entre les applications. Un de ces paradigmes qui a pris de l'ampleur au cours de ces dernières années est l'architecture orientée services (SOA). La méthode de conception des architectures SOA est basée sur un ensemble de standards permettant de créer une infrastructure informatique capable de répondre rapidement aux nouveaux besoins d'un utilisateur. Les services Web constituent un moyen de plus en plus normalisé, étendu et puissant pour mettre en œuvre une architecture SOA. Parmi les concepts intéressants qu'offre la technologie de service Web, il convient de souligner la possibilité de créer un nouveau service Web à valeur ajoutée par composition de plusieurs services Web existants. Il n'est pas toujours évident de trouver des services Web qui correspondent aux requêtes des utilisateurs. Par conséquent, la composition des services existants est un besoin grandissant de nos jours.

La composition de services implique la capacité de sélectionner, de coordonner, d'interagir, et de faire interopérer des services Web existants. Elle constitue une tâche complexe. Cette complexité est due principalement au grand nombre de services Web disponibles sur le Web et à leur hétérogénéité puisqu'ils sont créés par des organisations différentes [Bucchiarone et Gnesi, 2006].

Cette complexité est renforcée quand il s'agit d'intégrer dynamiquement des services à la demande, et surtout les composer pour répondre automatiquement à des exigences qui ne sont pas réalisées par les services existants. Ce défi est d'autant plus grand dans le cas d'un environnement, tel que le Web et l'informatique pervasive où les entités disponibles sont dynamiques et les attentes des utilisateurs sont variables et personnalisées. En effet, une approche pour la composition de services doit offrir le potentiel de réaliser des applications flexibles et adaptables, en sélectionnant et en combinant les services de manière appropriée sur la base de la requête et du contexte de l'utilisateur .

Dans cette perspective, différentes approches ont été développées pour la composition dynamique de services. Cependant, la plupart d'entre elles considèrent l'ensemble de ser-

vices Web comme le seul aspect configurable au moment de l'exécution. En conséquence, les processus de composition ont tendance à être statique et non-flexible dans le sens où ils n'ont pas la capacité de s'adapter aux besoins des utilisateurs.

Afin de construire une composition de services, deux étapes doivent être effectuées [Santos *et al.*, 2006] (pas nécessairement séparées) : un schéma de composition (ou modèle de processus) spécifiant le flux de contrôle et de données entre les activités doit être créé ; (2) les services concrets doivent être découverts et assignés aux activités du processus.

En ce qui concerne le degré de dynamicité dans ces deux étapes, nous avons retenu essentiellement deux stratégies de composition. La première consiste à définir le schéma de composition au moment de la conception et à sélectionner les services concrets au moment de l'exécution en se basant sur des critères automatiquement analysables, telles que la fonctionnalité du service, la signature et les paramètres QoS. [Casati *et al.*, 2000] utilise une méthode statique pour la génération de workflow. Un service composite est modélisé par un graphe qui définit l'ordre de l'exécution. Les services concrets peuvent être assignés aux tâches du workflow au moment de l'exécution. Dans Meteors [Aggarwal *et al.*, 2004], les auteurs ont proposé une approche qui consiste à ajouter de la sémantique aux standards courants tels que UDDI, WSDL et BPEL. Toutefois, ces deux initiatives nécessitent un workflow prédéfini. Par conséquent, une telle méthode de composition manque de flexibilité et présente des difficultés à s'adapter aux besoins des utilisateurs.

La deuxième stratégie consiste à combiner la génération du schéma de composition avec l'assignation des service au moment de l'exécution. Cela implique que la composition complète de services peut être effectuée lors de l'exécution. Généralement, les méthodes inspirées de l'intelligence artificielle (IA) basées sur la logique formelle sont utilisées dans le but de fournir une composition entièrement automatisée, comme le raisonnement automatisé par les preuves de théorèmes (theorem proving). Un exemple de composition de services entièrement automatisée basée sur des algorithmes de planification inspirés de l'IA est donnée dans [Ponnekanti et Fox, 2002]. Pour créer un service composite, le demandeur de services a besoin seulement de spécifier l'état initial et final pour le service composite, puis la génération de plan peut être obtenue en utilisant un système expert basé sur des règles. Une solution pour la composition automatique basée sur un langage de description de but (GDL4WSAC) est proposé par *Lin et. al.* [Lin *et al.*, 2005].

Bien que cette deuxième stratégie semble être plus flexible et adaptable que la première, la modélisation des flux de contrôle et de données d'un service composite est une tâche potentiellement fastidieuse et consommatrice en terme de temps (raisonnement, planification),

et surtout si nous considérons une composition totalement automatisée. En outre, la création du flux de données, peut être complexe et nécessite l'intervention de l'utilisateur censé avoir une connaissance approfondie sur les représentations de types sous-jacents. Donc même les approches qui clament fournir une composition « entièrement automatisée » au moment de l'exécution se basent sur une partie prédéfinie (comme les règles).

Dans notre travail, nous essayons de tirer profit des avantages des deux stratégies citées : en plus de la sélection des services concrets au moment de l'exécution, la génération du schéma de composition est effectuée en partie au moment de l'exécution en ayant recours aux services abstraits fournis au moment de la conception. Cela permet une certaine flexibilité et adaptabilité sans avoir besoin de construire une composition de services à partir de zéro au moment de l'exécution.

Dans cette même perspective, quelques travaux sur la composition des services ont utilisé une sorte de gabarit ou workflow flexibles. Le travail dans [Teije *et al.*, 2004] a utilisé une méthode à base d'agents pour dériver différents services complexes grâce à des templates. cependant, le seul aspect configurable est l'ensemble des services Web. Les auteurs dans [Wang *et al.*, 2010] modélisent une composition de services comme un processus de décision de Markov, de sorte que de multiples services et workflow peuvent être incorporés dans une composition de service unique. L'optimisation de la composition est réalisée au moment de l'exécution à travers l'apprentissage par renforcement. Lors de l'exécution de la composition, le système peut choisir le workflow qui offre les meilleurs résultats. Toutefois, l'accent dans ce travail est mis sur l'adaptation aux changements des services composants et non pas des exigences fonctionnelles des utilisateurs.

Dans cette thèse, nous proposons une approche qui se compose principalement de quatre étapes. La première étape prend une structuration des besoins de l'utilisateur matérialisée par un graphe d'intentions et l'enrichit pour expliciter les relations implicites. Le résultat de cette étape permet de générer un schéma de composition initial en construisant le flux de contrôle déduit du graphe des intentions enrichi, puis en sélectionnant les services abstraits adéquats. Le choix de ces services est basé sur le matching sémantique et le degré d'affinité entre les services abstraits. La troisième étape consiste à générer le schéma de composition final à l'aide d'un mécanisme de raffinement des services abstraits en utilisant des techniques de matching sémantique. Enfin, le plan d'exécution est généré en tenant compte des contraintes non-fonctionnelles fournies par la spécification des intentions.
Le reste de ce manuscrit est organisé comme suit :

Dans le chapitre 1, nous présentons un état de l'art qui est constitué de trois parties. La

première partie introduit la notion de services et présente les concepts fondamentaux de l'architecture orientée services. Nous exposons dans la deuxième partie les stratégies de composition des services en mettant l'accent sur l'importance de la composition automatique et dynamique des services. Ensuite, nous passons en revue les différents points d'intérêt de la composition automatique à savoir : la description de services, le matching, la sélection et la combinaison de services. Étant donné que le travail présenté dans ce manuscrit s'intéresse particulièrement à l'élaboration d'une composition automatique de services, la dernière partie de l'état de l'art est dédiée à l'étude des approches existantes de composition automatique de services. Nous nous focalisons surtout sur celles utilisant la sémantique. A la fin de ce premier chapitre, nous exposons la problématique dégagée suite à notre étude.

Dans le chapitre 2 nous présentons la première contribution de cette thèse qui consiste à l'élaboration des modèles des intentions et des composants utilisés dans la composition appelés services abstraits. D'abord, nous introduisons notre approche de composition guidée par les intentions de l'utilisateur et basée sur la sémantique des services. Ensuite, nous détaillons les modèles sémantiques du service abstrait et des intentions qui permettent d'avoir des matching sémantiques adéquats lors de la génération du schéma de composition.

Le chapitre 3 détaille la deuxième contribution de cette thèse qui est le processus automatique de composition ; il est divisé en quatre étapes qui prennent un graphe des intentions en entrée et génèrent un plan d'exécution comme résultat. Les aspects relatifs au choix des services, leur connexion et leur raffinement sont détaillés.

L'étape de la validation de notre travail est présentée dans le chapitre 4. Nous commençons tout d'abord par montrer l'architecture de l'environnement de composition. Puis, nous présentons les choix techniques et logiciels que nous avons faits pour l'implantation de nos contributions théoriques. Nous passons ensuite à l'application des différentes étapes du processus de composition sur un cas d'étude qui fait partie de la gestion d'un bâtiment intelligent.

Le document se termine par une conclusion générale qui présente une synthèse de nos contributions et qui expose les limites et les directions de recherches que ce travail a permis d'identifier.

1 État de l'art

1.1 Introduction

DANS ce chapitre, nous introduisons dans un premier temps la notion de services et nous présentons les concepts fondamentaux de l'architecture orientée services. Nous exposons dans un deuxième temps les stratégies de composition des services en mettant l'accent sur l'importance de la composition automatique et dynamique des services. Ensuite, nous passons en revue les différents points d'intérêt de la composition automatique à savoir : la description de services, le matching, la sélection et la combinaison de services. La dernière partie de l'état de l'art est dédiée à l'étude des travaux relatifs à la problématique de composition automatique de services. Avant de conclure, nous exposons la problématique dégagée suite à notre étude.

1.2 Concepts de base du paradigme orienté service

1.2.1 Évolution vers les approches orientées services

En génie logiciel, un certain nombre d'approches ont été proposées pour le développement de logiciels. Chaque nouveau paradigme proposé essaie de résoudre les limitations des paradigmes précédents en essayant de réutiliser certains principes et ajouter de nouveaux. C'est ainsi que les approches orientées objets [Taylor, 1998], orientées composants [Szyperski, 1998] et dernièrement les approches orientées services ont vu l'apparition.

Pour dissimuler la complexité d'intégration d'applications autonomes et hétérogènes, trois architectures à base de composants ont été proposées [Vinoski, 2004], [Schantz et Schmidt, 2002] : EJB (Enterprise Java Beans), CORBA (Common Object Request Broker Architecture), et .NET. Cependant, le couplage entre les objets est relativement fort dans ces technologies à base de composants distribués. Par exemple, la norme CORBA permettant la manipulation des objets à distance avec n'importe quel langage, nécessite une connaissance

de la structure des objets par les applications clientes et par les fournisseurs. Ceci implique que les différents partenaires doivent définir au préalable des règles de transformation objets/messages. En outre, on ne peut assembler que des objets CORBA (ou COM) entre eux, puisque chaque architecture propose sa propre infrastructure. Par conséquent, la mise en œuvre de ces architectures dans le cadre d'une infrastructure ouverte telle que Internet soulève des difficultés. Pour palier les limites de toutes ces architectures, les efforts de conception ont donné lieu au concept d'architecture orientée service (*Service Oriented Architecture SOA*)[Papazoglou *et al.*, 2008]. Ainsi, les approches orientées services ont vu le jour.

Le paradigme orientée services [1] est un paradigme interdisciplinaire destiné pour les applications distribuées. Son apparition a introduit une nouvelle manière dont les applications logicielles sont conçues, intégrées, fournies et utilisées. Les approches orientées services [Huhns et Singh, 2005] mettent en avant l'idée de composer des services indépendants pour réaliser des applications logicielles agiles faiblement couplées. Ces approches exploitent le concept de service comme élément fondamental autour duquel les applications complexes sont développées. L'idée de base est d'encapsuler les fonctionnalités offertes par les organisations sous forme de services. Par conséquent, l'intégration et la gestion des applications à base de services se focalisent au niveau des fonctionnalités sans tenir compte des technologies utilisées et en cachant les détails de l'implémentation.

Les approches à base de services offrent la possibilité de réaliser une interopérabilité à grande échelle en garantissant la souplesse nécessaire afin de s'adapter à l'évolution des technologies et des besoins des utilisateurs (individus ou entreprises). Un des principaux bénéfices des approches à base de services est le couplage faible entre le fournisseur et le consommateur de service d'une part, et entre les différents services réunis dans une même application d'une autre part. le consommateur d'un service n'a pas besoin d'avoir connaissance des détails techniques tels que la technologie d'implémentation et la plateforme d'exécution d'un service [Dustdar et Papazoglou, 2008b]. Ce faible coupage permet le développement d'applications de façon parallèle et indépendante, ce qui entraine une réduction des coûts de développement et d'intégration des applications.

1.2.2 Les services

Le service représente la brique de base dans une architecture orientée services. Il est assez difficile de donner une définition précise au service car il est utilisé dans plusieurs domaines. Il peut être défini tout simplement comme suit : un service est une action réalisée par une

1. Service-Oriented Computing

entité au profit d'une autre. De manière intuitive, la notion de service correspond à une abstraction des fonctionnalités d'une entité. Cette entité peut être simple comme une donnée ou un objet sur le web. Elle peut aussi être complexe comme un système d'information adaptable d'une entreprise. Avant l'utilisation d'un service, il est nécessaire de connaitre ses fonctionnalités qu'il accomplit et même ses caractéristiques non-fonctionnelles comme sa performance ou sa disponibilité. Plus particulièrement, un service logiciel (qui représente une entité logicielle mise à la disposition d'autres applications) doit être décrit et publié. Ses capacités sont ensuite découvertes par les utilisateurs éventuels qui peuvent par la suite exécuter le service pour obtenir la fonctionnalité demandée.

Étudions maintenant certaines définitions précises qui ont été données pour le service. Selon [Dustdar et Papazoglou, 2008b], « *Services are self-contained processes deployed over standard middleware platforms, e.g., J2EE, or .NET that can be described, published, located (discovered), and invoked over a network... Services are most often built in a way that is independent of the context in which they are used. This means that the service provider and the consumers are loosely coupled.* ». Cette définition présente le service comme un processus autonome qui est décrit, publié, découvert et invoqué via le réseau. La conception du service est indépendante de sa technologie d'implémentation et sa plate-forme d'exécution. En plus, le service ne connaît pas le contexte dans lequel il va être utilisé. Cette indépendance caractérise fortement les services et elle facilite le faible couplage. Ceci permet d'utiliser des services réalisés avec des technologies d'implémentation différentes et de les déployer sur des plates-formes hétérogènes.

OASIS, un consortium mondial travaillant sur le développement, la convergence et l'adoption de standards pour les applications informatiques, propose la définition suivante : « *A service is a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description... A service is accessed by means of a service interface where the interface comprises the specifics of how to access the underlying capabilities.* » [Brown *et al.*, 2006]. Cette définition présente un service comme un mécanisme permettant l'accès à une ou plusieurs fonctionnalités et dont l'accès est offert grâce à une interface définie. Cette interface décrit les fonctionnalités offertes par le service et un ensemble de contraintes et de politiques d'accès aux fonctionnalités offertes.

Arsanjani définit un service comme suit : « *A service is a software resource (discoverable) with an externalized service description. This service description is available for searching, binding, and invocation by a service consumer.* » [Arsanjani, 2004]. Dans cette définition, les principales interactions permettant l'utilisation des fonctionnalités d'un service sont identifiées. Un client

de service passe par une étape de recherche pour découvrir la description du service correspondant à ses critères. Ensuite, cette description permet de faire la liaison avec le service et de réaliser l'invocation du service.

[Han *et al.*, 2009] considère un service comme une abstraction du niveau métier implémentant les processus métiers. Une définition du concept service est énoncée par [Cauvet et Guzelian, 2008], qui considère un service comme une unité réutilisable encapsulant un ou plusieurs fragments d'un processus métier et visant à satisfaire des buts métiers.

A travers les différentes définitions, nous ressortons une idée principale, à savoir qu'un service permet d'exposer une ou plusieurs fonctionnalités décrites par une description de service. Cette description est utilisée par les clients du service pour rechercher, sélectionner et invoquer le service adéquat. Un ensemble de caractéristiques spécifiques aux services peuvent être distinguées [Bachtarzi, 2014]

- Les services sont réutilisables puisque chaque fonctionnalité de service peut être utilisée plusieurs fois.
- Les services sont composables. En effet, un nouveau service peut être créé par composition de plusieurs services existants. La composition peut être vue comme une forme de la réutilisation puisqu'un service particulier peut participer dans plusieurs compositions.
- Les services sont faiblement couplés, ce qui permet de réduire les dépendances et d'assurer une meilleure flexibilité.
- Les services peuvent être vus comme des boites noires puisqu'ils ne sont accessibles qu'à travers leurs interface cachant les détails d'implémentation des fonctionnalités fournies.

Le service est l'élément de base de l'architecture orientée service introduite ci-après.

1.2.3 Architecture orientée service

Les architectures orientées services (SOA) présentent un style d'architecture permettant de définir les interactions entre le service et ses utilisateurs. Différentes définitions de l'architecture orientée services ont été publiées. La définition proposée par Microsoft est la suivante : « *The policies, practices, frameworks that enable application functionality to be provided and consumed as sets of services published at a granularity relevant to the service consumer. Services can be invoked, published and discovered, and are abstracted away from the implementation using a single, standards-based form of interface.* » [Sprott et Wilkes, 2004]. Ainsi, une architecture orientée services est l'ensemble des politiques, des pratiques et des environnements qui

permettent d'intégrer différentes fonctionnalités exposées comme des services.

[Dodani, 2004] considère qu'une architecture orientée services fournit le moyen pour une intégration flexible des applications et des ressources. Ceci est du (1) au fait que chaque application ou ressource est représentée en tant que service avec une interface normalisée, (2) à la possibilité d'échanger des informations structurées (messages, documents, objets métier) entre les services, et (3) à la coordination des services pour qu'ils puissent être invoqués et utilisés de manière efficace par les utilisateurs ou par d'autres services.

Le modèle de référence d'OASIS [Brown *et al.*, 2006] définit l'architecture SOA comme « un paradigme permettant l'organisation et l'utilisation des services distribuées qui peuvent être sous le contrôle de différents auteurs. SOA fournit un moyen uniforme pour offrir, découvrir, interagir avec et utiliser les services pour produire des résultats désirés en conformité avec les conditions et les attentes mesurables ».

En considérant les différentes définitions, nous pouvons constater que la plupart d'entre elles affirment que l'architecture orientée services est un style architectural qui permet la réorganisation et le redéploiement du système d'information. En effet, l'architecture orientée services permet l'encapsulation des fonctionnalités fournies par un système d'information en un ensemble de services faiblement couplés. Les services sont munis d'un contrat d'utilisation et d'une interface de description. Ainsi, l'architecture orientée services a introduit une nouvelle philosophie pour le développement des applications distribuées, où les services peuvent être publiés, découverts, composés, réutilisés, et invoqués en utilisant l'interface, indépendamment de la technologie utilisée pour implémenter chaque service [Granell *et al.*, 2010].

Dans une architecture orientée services, l'utilisation des services suit un protocole bien établi : publication, découverte et invocation. La figure 1.1 [Brown *et al.*, 2006] présente les principaux acteurs qui interviennent dans une architecture orientée services : fournisseur du service, annuaire de services et consommateur de services.

Le fournisseur de services désigne l'entité propriétaire du service. Il a pour rôle de développer les fonctionnalités du service, de générer sa description et de le déployer. La description de ce service est publiée dans un annuaire afin que les consommateurs de services puissent découvrir et accéder au service. Le consommateur du service correspond au demandeur du service. Il s'agit d'une application cliente ou un autre service qui interroge l'annuaire de services pour invoquer le service qui correspond à ses besoins. La spécification du service disponible dans l'annuaire est utilisée pour découvrir le service adéquat et établir une connexion avec le fournisseur. L'annuaire donc joue le rôle d'intermédiaire entre

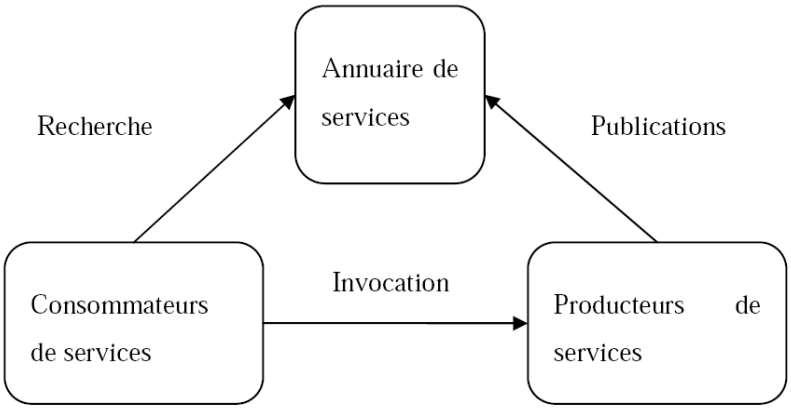

Figure 1.1 — Architecture orientée services

les clients et les fournisseurs de services. L'ensemble de ces interactions entre les consommateurs et les fournisseurs de services sont effectuées au sein d'une architecture orientée services à travers un environnement d'intégration et d'exécution de services.

En termes de technologies, les architectures orientées services ne sont pas étroitement liées à un éditeur ou une implémentation particulière. Cependant la technologie d'implémentation des SOA la plus connue et qui a pris de plus en plus d'ampleur est celle basée sur les services web. Ceux-ci sont des composants logiciels qui reposent sur le Web comme infrastructure pour la gestion de la communication entre les différents composants. Le grand intérêt porté aux services web a abouti à une petite confusion qui est de rendre synonyme la notion de services web avec la notion de services issu de l'architecture orientée services. Les services web constituent simplement une implémentation particulière des principes des SOA. Ils se caractérisent par une standardisation des implémentations, par une localisation à distance des services et par une récupération de l'interface d'accès permettant l'exécution du traitement correspondant. En effet, les services web reposent sur des technologies ouvertes basées sur des standards XML (eXtensible Markup Language). Ils offrent ainsi une solution indépendante des langages de programmation, des systèmes d'exploitation et de matériel spécifique. Les trois protocoles de base sont WSDL [Christensen *et al.*, 2001], UDDI [Committee, 2004] et SOAP [Mitra et Lafon, 2007]. La description de l'interface d'un service Web est effectuée par le WSDL (web Service Description Language) qui repose sur la notation XML. La description d'un service avec WSDL doit inclure l'emplacement du service web ainsi que les opérations (méthodes, paramètres et valeurs de retour) que le service propose. Le standard UDDI (Universal Description Discovery and Integration) constitue une norme pour les annuaires de services Web. Il vise à décrire une manière standard de publier les

services web au sein d'un service d'annuaire. Il offre par ailleurs une API aux applications clientes, pour consulter et extraire des données concernant un service et son fournisseur. Le SOAP (Simple Object Access Protocol) est un protocole défini à l'origine par Microsoft, puis standardisé par le W3C. Sa spécification utilise la notation XML permettant de définir les mécanismes d'échanges d'information entre des clients et des fournisseurs de services web.

1.3 La composition de services

L'apparition des applications à base de services a ouvert de nouvelles opportunités de développement d'applications par composition de services. La composition de services fait référence au processus de combinaison de fonctionnalités de multiples services en un seul service composite à valeur ajoutée. Ce dernier offrirait une fonctionnalité qui ne pourrait pas être fournie par un service unique existant. La composition de services est un sujet de grand intérêt autant pour le monde de la recherche que pour le monde industriel. De nombreux travaux de recherche visent à développer des modèles de composition de services et à fournir les outils nécessaires pour la composition.

L'ensemble du processus de composition de services doit être considéré plus largement que le problème de combinaison des services. Il comprend également le problème de la description des buts de l'utilisateur, de l'acquisition des données nécessaires de l'utilisateur, de la conception et l'exécution du meilleur service composite possible, et de la présentation des résultats à l'utilisateur [Yang et Papazoglou, 2004]. Comme le montre la figure 1.2, l'ensemble du processus intègre deux parties : l'utilisateur et le système de composition.

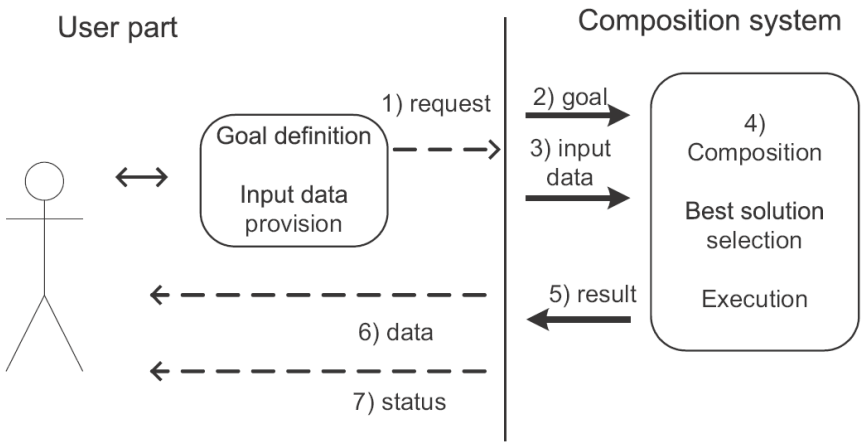

Figure 1.2 — Processus de composition des services [Bartalos et Bieliková, 2011]

L'utilisateur, qui peut être humain ou un système logiciel, envoie une requête (1). Celle-ci

contient la description du but (2). L'utilisateur doit également fournir des données d'entrée (3). Ceux-ci peuvent être définis par l'utilisateur lui même, ou le système de composition identifie quelles données sont nécessaires. Après ces étapes, la conception du workflow et l'exécution ont lieu (4). En général, le but de l'utilisateur peut être réalisé par plusieurs solutions alternatives. Cela varie selon la considération des questions qui peuvent être divisées en deux groupes. Le premier comprend les questions touchant le degré de satisfaction des utilisateurs en fonction de la qualité du résultat, obtenu par l'exécution du service composite, par exemple l'utilisateur peut avoir des préférences. le second concerne la qualité du processus de l'exécution affectée par les attributs de la qualité de service (QoS) qui sont des propriétés non-fonctionnelles des services. L'objectif est de trouver la meilleure solution tenant compte de toutes ces questions. Habituellement, il n'y a pas de solution étant la meilleure de chaque point de vue. La solution retenue est celle qui a le meilleur score global. La solution trouvée est exécutée pour produire le résultat requis (5). Si l'utilisateur a demandé des données, celles-ci sont récupérées à partir du résultat fourni (6). L'utilisateur est également informé de l'état résultant de l'exécution, par exemple s'il a réussi (7).

Généralement, le but désiré par l'utilisateur ne peut être satisfait par l'exécution d'un service unique. Mais plutôt, un workflow plus complexe est conçu, dans lequel l'exécution de services web se présente comme une activité atomique. Les services sont combinés en utilisant des structures de contrôle tels qu'une exécution parallèle ou séquentielle. La combinaison de services est nécessaire pour deux raisons. Tout d'abord, le but de l'utilisateur peut être constitué de plusieurs sous-buts, chacun réalisé par un service différent. Deuxièmement, l'exécution de certains services pourrait être nécessaire, pour produire des données ou obtenir des effets qui sont nécessaires pour exécuter d'autres services. Le mécanisme derrière la composition de services web automatique doit concevoir un schéma décrivant le flux de contrôle et de données de l'exécution, c'est à dire qu'il représente la synchronisation des exécutions de services particuliers, et prescrit quel service produit des données de sortie utilisées comme entrées par d'autres services.

1.3.1 Stratégies de composition de services

La composition de services est classée en fonction de deux aspects principaux [Dustdar et Papazoglou, 2008a]. Tout d'abord, nous distinguons entre la composition statique et dynamique. Deuxièmement, nous divisons les approches de composition entre manuel, et automatique.

Le premier point de vue concerne le moment où le service est sélectionné pour être un

composant du service composite. Dans les approches statiques, la sélection est faite au moment de conception. Les approches dynamiques effectuent la sélection des services au cours de l'exécution, tenant compte des besoins exprimés dans le but d'un utilisateur particulier.

Le deuxième aspect porte sur le fait si la composition est faite manuellement par un concepteur, ou automatiquement par un système de composition.

1.3.2 Composition statique vs composition dynamique

Dans les environnements de composition, les approches de composition des services web peuvent être classées en deux catégories selon le moment de la conception et de l'exécution au cours desquels les services web sont composées et exécutées, à savoir composition statique et composition dynamique [Jiang et Bai, 2013]. La différence principale entre les compositions statiques et dynamiques est le temps pendant lequel un service web concret est intégré dans la composition. Les approches statiques sélectionnent les services au moment de conception ; tandis que les approches dynamiques sélectionnent les services au cours de l'exécution. Selon [Foster, 2004], « *Static web service compositions are known at design time and are bound to a composition at design time. Dynamic web service compositions are one or many compositions in which web services are not known at design time, and which are discovered or their properties resolved based upon a criteria process set at run time.* »

Les environnements de composition dans lesquels les services web sont composés et exécutés, peuvent également être divisés en deux catégories, à savoir, les environnements de composition statiques et les environnements de composition dynamiques [Groba et Clarke, 2014], selon les informations contextuelles, qui peuvent influencer le processus de composition de services et/ou l'exécution des services composites. L'information contextuelle désigne toute information, événement, variation ou changement qui peuvent survenir au moment de conception ou au moment de l'exécution, tels que la disponibilité des services web, les besoins de composition, des changements dans la qualité de service, (par exemple, le prix, la réputation ,etc), la dégradation des ressources système (CPU, stockage de mémoire et bande passante).

Les approches de composition dynamiques doivent être en mesure d'adapter le processus de composition et d'exécution des services web aux changements et à l'évolution de l'environnement et de chercher à minimiser les interventions de l'utilisateur afin de fournir les services composites les plus appropriés et satisfaire les besoins des utilisateurs.

La plupart des approches de composition fonctionnent correctement dans les environnements de composition statiques où les services web impliqués dans les compositions ou

les informations contextuelles telles que les exigences changent rarement. Cependant, les compositions statiques ne sont pas flexibles dans le sens où ils ne sont pas adaptables aux changements en temps d'exécution tels que : les fournisseurs de services publient de nouveaux services, ou des services sont remplacés par d'autres. Dans ce cas, il est inévitable de reconstituer le service composite. La composition de services dynamique vise à surmonter les problèmes qui se manifestent dans les compositions statiques. Son but est de composer des services complexes à la volée, en tenant compte des informations contextuelles.

La plupart des approches existantes de composition de services dynamique s'appuient sur des plans de composition abstraits prédéfinis et effectuent la sélection de service au moment de l'exécution. Dans ce modèle, les processus de composition et d'exécution des services web sont entrelacés afin de fournir à la volée des solutions de composition. Un plan de composition abstrait comprend un certain nombre d'actions (généralement représentées comme des activités abstraites). Il indique l'ordre d'exécution approprié des différentes activités. La phase de sélection vise à sélectionner un service concret pour chaque activité et à les intégrer dans le plan de composition au moment de l'exécution pour construire un service composite spécifique.

Par conséquent, le seul aspect pris en considération dans la plupart des approches de composition dynamiques (comme par exemple dans [Mabrouk *et al.*, 2015], [Alrifai *et al.*, 2010], [Akzhalova et Poernomo, 2010], [Yu *et al.*, 2007], [Canfora *et al.*, 2005a]) est la qualité de service (QoS). D'une autre part, ces approches ne peuvent pas faire face à certains cas où le plan de composition ne peut être que partiellement défini ou ne peut pas être défini à l'avance. La gestion de crise (crisis management) est un exemple des applications métiers où les crises se produisent souvent brusquement, de sorte que les bonnes actions ne peuvent pas être précisément identifiées avant qu'une crise se produit et l'information contextuelle ne peut pas être identifiée à l'avance (par exemple, les témoins , les rapports de police, les conditions météorologiques, etc).

1.3.3 Composition manuelle vs automatique

Un service web composite offre des fonctions plus complexes que celles fournies par un service unique en combinant plusieurs services. Cependant, comme le nombre de services web augmente et les requêtes des utilisateurs sont de plus en plus complexes, la tâche de trouver les services souhaités devient de plus en plus difficile et prend beaucoup de temps. En outre, il devient de plus en plus difficile de construire manuellement un service web composite.

Les premiers travaux ont contribué essentiellement à des approches de composition de services manuelles et semi-automatiques, tandis que la plupart des recherches récentes se concentrent à automatiser les processus de composition de services web.

La composition de services manuelle est basée sur l'intervention humaine. La plupart des approches de composition manuelle [Taylor *et al.*, 2003] [Benatallah *et al.*, 2003] laissent à la charge des utilisateurs la génération des workflows spécifiques et exécutables pour représenter des services composites et décrire leur ordre d'exécution, soit graphiquement ou à travers des langages déclaratifs. Afin de créer manuellement des workflows ou des processus basés sur des services, de nombreux langages de modélisation de processus ont été proposés tels que BPMN [Skersys *et al.*, 2012] , BPEL [OASIS Web Services Business Process Execution Language (WSBPEL) Technical Committee, 2007], etc.

Rappelons que, pour construire une composition de services, il faut définir un schéma de composition (ou modèle de processus) spécifiant le flux de contrôle et de données entre les activités et assigner les services concrets aux activités du processus. Le degré d'automatisation dans la création du schéma de composition est une caractéristique importante d'une stratégie composition de services. Les méthodes de composition de service traditionnelles requièrent de l'utilisateur de définir le flux de données et le flux de contrôle manuellement, soit directement, soit par l'intermédiaire d'outils de conception. En considérant la multitude de services disponibles, sélectionner manuellement les services adéquats pourrait être une tâche fastidieuse. En outre, la création de flux de données est une tâche complexe et exige que le compositeur ait une connaissance approfondie sur les représentations et la signification des paramètres sous-jacents. Des stratégies de composition avancées soutiennent activement l'utilisateur dans la création du schéma de composition. Elles sont souvent considérées comme des méthodes de composition semi-automatiques. Elles permettent de faciliter et d'accélérer la création du schéma de composition. Les approches de composition entièrement automatiques visent la génération d'un plan de composition de services sans interaction humaine. La plupart des méthodes de composition inspirées de l'intelligence artificielle et basées sur la logique formelle sont utilisées pour cette fin. Au moyen d'un algorithme de planification, un workflow contenant des activités ou des services concrets est généré pour satisfaire aux exigences établies par le demandeur. S'il y a plusieurs solutions pour le problème, à savoir plusieurs plans satisfaisant une requête, une sélection est effectuée sur la base des paramètres de QoS. La combinaison de la génération du schéma de composition avec la sélection des services concrets d'une manière automatique implique que la composition complète de services peut être effectuée au moment de l'exécution. La question de « jusqu'à quel point le processus de composition peut être automatisé » est l'objet de nom-

breux travaux.

1.3.4 Degrés d'automatisation/dynamicité

Le degré d'automatisation et de dynamicité dans les approches de composition peut être utilisé pour la classification des stratégies existantes de composition de services selon six catégories, comme le montre la figure 1.3.

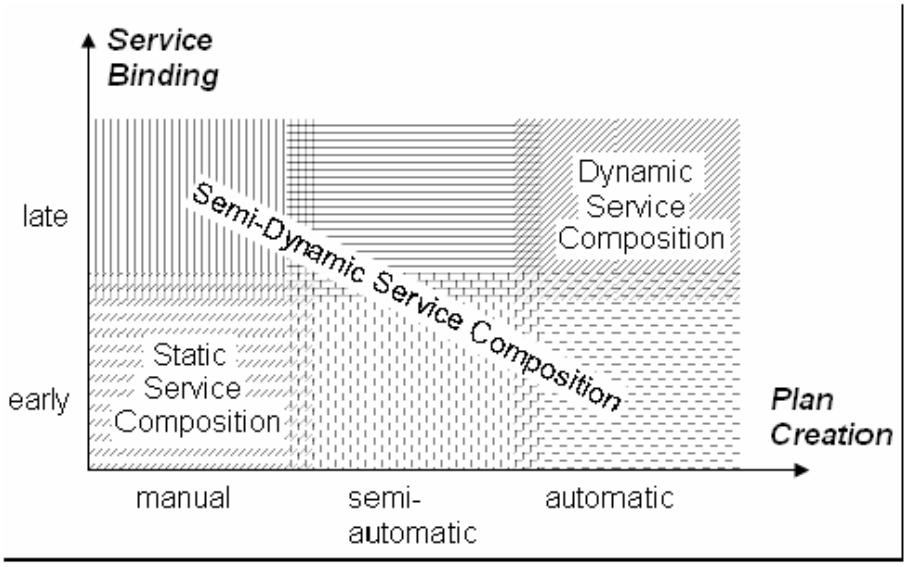

Figure 1.3 — Classification des stratégies de composition des services [Santos *et al.*, 2006]

Le fait que les frontières entre ces catégories ne sont pas strictes apparaît clairement à travers des transitions progressives (en douceur) entre les carrés. Certaines catégories peuvent se chevaucher, c'est à dire, il y a des approches de composition qui peuvent être attribuées à deux ou plusieurs catégories. Pour donner un exemple : Outre le binding précoce et tardif (*early and late binding*) il peut y avoir plusieurs variations entre les deux, comme la spécification d'un ensemble restreint de services concrets candidats au moment de conception à partir desquels un service est choisi et invoqué au moment de l'exécution. Signalons ici que le terme « *late binding* » est souvent utilisé pour indiquer la sélection de services web basée sur les propriétés non-fonctionnelles au moment de l'exécution et avant l'invocation des services web constituants [Châtel *et al.*, 2010].

Il existe de nombreux exemples pour chacune des catégories identifiées. Dans EFlow [Casati *et al.*, 2000] et les systèmes similaires, le flux de données et le flux de contrôle d'un service composite sont définies manuellement. Les services, toutefois, peuvent être sélectionnés au début lors de la conception ou au moment de l'exécution. Un exemple de création

semi-automatique du modèle de processus en se basant sur des descriptions sémantiques du service a été présenté par Sirin et. al. [Sirin *et al.*, 2002]. Tous les services possibles qui correspondent à l'activité en cours sont présentés à l'utilisateur. L'approche de composition semi-automatique décrite par Fluegge et Tourtchaninova [Flügge et Tourtchaninova, 2004] utilise des blocs fonctionnels abstraits qui sont liés à des services concrets à l'exécution. L'utilisateur est soutenu pour la création du modèle de processus en analysant automatiquement les concepts des inputs et outputs de ces blocs. L'objectif du travail présenté dans [DiBernardo *et al.*, 2008] est de réduire le nombre de choix que les utilisateurs ont à faire en restreignant l'ensemble de services Web en classant les services Web de sorte que les plus appropriés sont présentés en premier lieu. Dans [Zahoor *et al.*, 2009], les auteurs proposent une approche de composition semi-automatique à base de règles, donnant l'utilisateur final le contrôle pour guider le processus de composition. Les utilisateurs finaux sont capables de construire le flux de composition en sélectionnant les services web représentés sous forme de nœuds. Ils les connectent ensuite en utilisant un ensemble de connecteurs de flux de contrôle et de données. Un exemple de composition automatisée de services basée sur des algorithmes de planification inspirées de l'intelligence artificielle (IA) est donnée par Ponnekanti et Fox [Ponnekanti et Fox, 2002]. Une approche pour la composition automatique basée sur un langage de description de but (GDL4WSAC) est proposé par Lin et. al. [Lin *et al.*, 2005].

Les approches de composition dynamiques sont plus adaptées à un environnement dynamique. En outre, les problèmes de performance peuvent également représenter un problème, car les processus de découverte et de matching de services ainsi que l'application des algorithmes sophistiqués de l'IA pour la génération automatique de plans peuvent être des tâches fastidieuses et consommatrices en termes de temps. Il est important qu'au moment de l'exécution, la réaction à une requête se fasse le plus rapidement possible. C'est assez difficile à réaliser par la plupart des approches de composition automatiques puisqu'elles visent à concevoir et composer un service composite à partir de zéro (*from scratch*), comme concrétiser tout un workflow abstrait ou générer un workflow en reliant les services un par un. C'est difficile de réaliser cette tâche en temps réel puisque certaines tâches, comme par exemple le raisonnement sémantique à travers une ontologie de domaine, consomment beaucoup de temps.

1.3.5 Points d'intérêt de la composition de services

La composition automatique de services est un sujet académique d'une grande importance, impliquant et profitant de plusieurs thématiques du domaine informatique (web sémantique, intelligence artificielle, etc). Le sujet implique plusieurs branches telles que la découverte de services, la description des processus et le matching sémantique. Il est assez difficile de partitionner clairement ces branches en sous-thèmes isolés puisqu'ils présentent des parties enchevêtrées et entrelacées les unes aux autres. Par exemple, la façon dont les services sont décrits et exprimés influence largement la manière de matcher, classer, combiner, et sélectionner les services.

Dans cette section, nous présentons les points d'intérêt de la composition automatique à savoir : description de services, matching, sélection et combinaison de services. Avant de commencer la discussion pour chaque point, un élément crucial dans le domaine de recherche de la composition automatique devrait être brièvement présenté puisqu'il est impliqué largement dans ce sujet : l'ontologie. Le terme ontologie est à l'origine de la philosophie, et en informatique , elle est utilisée pour modéliser et représenter les connaissances dans un domaine spécifique. En capturant les concepts d'un domaine et les relations entre eux, elle peut faire du raisonnement sémantique et logique. Gruber proposait la définition la plus citée. Il définit l'ontologie comme étant une spécification explicite d'une conceptualisation [Gruber, 1993]. La conceptualisation est le résultat de l'analyse du domaine de l'étude et l'abstraction du monde de ce domaine. Une forme concrète peut être utilisée pour représenter cette conceptualisation. En effet, les concepts, les relations ainsi que les contraintes sont explicitement définis dans un format et un langage formel. Cette conceptualisation a été par la suite raffinée dans le domaine de l'informatique par Neches et al [Neches *et al.*, 1991]. Ces auteurs définissent l'ontologie comme étant les termes et les relations de base comportant le vocabulaire d'un domaine et les règles pour combiner les termes et les relations afin de définir des extensions du vocabulaire.

En considérant le domaine de la composition automatique, [Syu *et al.*, 2012] ont identifié que, dans ce domaine, l'ontologie a deux objectifs principaux. Le premier consiste à améliorer les services ayant une description syntaxique en les transformant en des services sémantiques ayant une interface exploitable par des machines, ce qui facilite l'automatisation des tâches de découverte et de composition des services. Un autre but est de générer un workflow en utilisant la connaissance du domaine modélisée à travers l'ontologie. Nous allons les expliquer dans les sections concernant la description des services et la combinaison des services, respectivement.

1.3.5.1 Description de services

Comme expliqué ci-dessus, la description du service influence largement les autres points d'intérêt de la composition de services. Dans les architectures SOA, les services sont réutilisables et sont considérés comme boîte noire. La seule façon de comprendre les services est via leurs descriptions et spécifications externes. Pour la description de service, actuellement le standard de facto et qui est le plus largement accepté et utilisé est WSDL. Cependant, WSDL est à peine suffisant pour générer des interfaces syntaxiques pour connecter et invoquer des services. La description du service reste uniquement au niveau fonctionnel, c'est-à-dire qu'elle contient la manière dont on peut utiliser le service et non ce que fait le service. Par conséquent, la description WSDL reste insuffisante lors du processus de sélection. Des travaux proposent des moyens de décrire des services web sémantiques afin de pallier cette difficulté et pour mettre en œuvre d'autres aspects tels que la découverte automatique de services. Afin d'automatiser les processus de recherche (action issue de la requête du client) et de sélection des services web, des travaux académiques ont été initiés, principalement dans le domaine du web sémantique [Payne et Lassila, 2004], [Hong *et al.*, 2009], [Sharifi *et al.*, 2011], [Peng, 2013]. Les services web issus des travaux de ce domaine sont appelés des services sémantiques. Les services web sémantiques sont des services web dont la description est améliorée par des langages empruntés au web sémantique, tel que RDF [Klyne et Carroll, 2004] et OWL. Cet emprunt au web sémantique permet à ces services d'être découverts et sélectionnés automatiquement par des machines ou d'autres services web distants. Ceci permet aux services web sélectionnés de répondre au mieux à la requête du client.

Plusieurs travaux ont proposé des langages de description de services web sémantiques comme OWL-S [Martin *et al.*, 2004], WSMO, et SAWSDL [Kopecký *et al.*, 2007]. En utilisant une ontologie de domaine appropriée définissant les termes du domaine de l'application, ces langages permettent la manipulation automatique des services. Pour simplifier, les langages de description représentent généralement les services en termes de tuple. Par exemple, WSDL décrit simplement un service (opération) en termes d'entrées et de sorties (IO), à savoir les messages SOAP destinés et produits par le service. Avec une interface sémantique, le tuple pourrait être plus riche, en incluant : Entrées, Sorties, Préconditions, Effets (IOPE : Inputs/Outputs/Preconditions/Effects), capacité, et les propriétés non-fonctionnelles. Celles-ci sont des attributs du service tels que la qualité de service (QoS). En outre, comme l'a mentionné [Zhao *et al.*, 2009] , une tendance récente, QoSaware WS, est largement adoptée. Par conséquent, la spécification sémantique pour les attributs

des qualités de service est nécessaire. Dans [Zhao *et al.*, 2009], un schéma d'ontologie globale a été défini pour décrire sémantiquement les classes, les QoS, et la connaissance du domaine des services. L'ontologie définissant les termes concernant les caractéristiques non-fonctionnelles des services serait plus importante lorsque l'on considère de plus en plus des critères de sélection de services.

1.3.5.2 Matching des services

Le matching est l'opération de correspondance entre deux concepts selon certains critères de similarité. Comme mentionné précédemment, puisque le tuple est l'idéal façon de comprendre et de décrire les services, le matching de services s'intéresse à la similarité fonctionnelle et le calcul de la compatibilité entre les tuples. Nous avons identifié que le matching de services a généralement deux types, qui sont conceptuellement équivalents. Ces deux types sont : matching entre services et matching des jonctions entre services. Plus précisément, le premier type consiste à évaluer la similarité entre un service et un service, entre un service et une activité abstraite, ou entre un service et une requête. Une activité abstraite représente une tâche qui a besoin d'être associé à un service pour la satisfaire. Ainsi, généralement une activité est également décrite par le tuple utilisé pour représenter le service (exemple : IOPE). En ce qui concerne le matching de requête, la manière la plus intuitive et la plus commune pour définir une requête suit l'approche « *query by example* » [Plebani et Pernici, 2009]. Ainsi, une requête est le plus souvent représentée par le même tuple qui représente le service. Par conséquent, que ce soit un service, une activité, ou une requête, ils peuvent être tous spécifiés de manière identique, ayant le même tuple. Par exemple, étant donné deux services représentés par le tuple (IO), il s'agit de calculer la similarité entre les valeurs des deux entrées et celle entre les valeurs de sorties. Le deuxième type de matching est entre les jonctions des service ; il s'agit de calculer la compatibilité entre l'Output d'un service (ou son Effet) et l'Input du service successeur (ou sa Précondition), le but étant de vérifier si deux services sont admissibles à être connectés ensemble. Le matching entre les jonctions de services joue un rôle important dans la combinaison de services comme nous allons le voir plus tard. Dans ce qui suit, nous allons nous concentrer sur la similarité et la compatibilité entre deux services. Pour la description de service, nous avons déjà dit que la tendance actuelle est d'utiliser des services sémantiques, où chaque service est décrit à l'aide d'une ontologie ou un tuple conceptuel. Cependant cette description seule reste insuffisante. Pour tenir compte de la sémantique des termes utilisés, chaque terme qui apparaît dans l'ontologie du service ou dans le tuple conceptuel doit être spécifié dans une ontologie modélisant le domaine en question [Lécué et Mehandjiev, 2009]. Comment définir précisément

ces termes est une autre question, parce que la conception de l'ontologie et l'annotation des termes sont assez coûteux [Ren *et al.*, 2011] [Belhajjame *et al.*, 2008]. La manière la plus primitive comme les travaux [Syu *et al.*, 2011] et [Xu *et al.*, 2010] utilise des mots-clés pour définir les termes. Le calcul du matching se fait lettre par lettre pour comparer les chaînes de caractères, ce qui est loin d'être suffisant. Une explication de l'insuffisance de l'utilisation des mots-clés ainsi que leurs inconvénients est présentée dans l'introduction de [Martin *et al.*, 2004]. Il existe plusieurs efforts qui tentent d'améliorer cette comparaison primitive. Tout d'abord, dans [Martin *et al.*, 2004], les auteurs ont proposé un algorithme de clustering des mots-clés ; ils sont associés à la signification sémantique, dans le même concept. Cependant, il n'est toujours pas aussi précis puisque les relations entre les concepts (ex. sous-concept) ne peuvent pas être reconnues par cet algorithme. Le travail présenté dans [Plebani et Pernici, 2009] exploite une base de données lexicale générale de WordNet (une ontologie générale de mots en anglais) pour raisonner sur la relation sémantique et la distance (similitude) entre deux termes. Poursuivant le même but de résoudre l'insuffisance de l'utilisation des mots clés dans le matching, les auteurs dans [Lu *et al.*, 2013] ont proposé une méthode de découverte de services Web qui combine l'utilisation de WordNet et des ontologies de domaine afin d'améliorer la qualité et la précision des services découverts en utilisant les technologies du web sémantique. Le travail présenté dans [Ren *et al.*, 2011] utilise WordNet pour spécifier les définitions sémantiques pour les termes. Par ailleurs, il a également proposé des formulations pour le calcul sémantique à travers des ontologies hétérogènes. Il y a plusieurs relations entre les termes annotés sémantiquement, tels que le matching exact, plugin, subsume, intersection, et disjoint [Lécué et Mehandjiev, 2009]. Un exemple de matching entre les jonctions de services peut être le suivant : une entrée contenant « *Food* » est compatible avec une sortie contenant « *Salad* » ou tout autre sous-concept de « *Food* ». Ainsi, il s'agit d'une connexion appelée lien de causalité (ou lien sémantique) [Lécué et Mehandjiev, 2009] [Lécué et Léger, 2006]. La relation n'est pas symétrique, par exemple, une entrée contenant « *Salad* » est incompatible avec une sortie contenant « *Food* ».

1.3.5.3 Combinaison de services

Le problème de combinaison des services est, dans notre opinion, au cœur du problème de la composition des services car elle définit la structure du processus (ou workflow). De ce fait, pour arriver à résoudre le problème de la composition, il faut résoudre principalement le problème de combinaison des services.

Bien que les solutions concernant la combinaison des services sont variées et disparates,

d'une manière générale, il y a deux différents mécanismes. Le premier mécanisme est de créer en quelque sorte un modèle de workflow abstrait et non-exécutable, puis transmettre le modèle aux approches relatives à la sélection de services. Le deuxième mécanisme est de synthétiser directement des services disponibles et connus en tant que service composite exécutable selon un workflow. Le premier mécanisme s'apparente à la méthode de composition *top-down* et le second mécanisme est similaire à la méthode *bottom-up*. Dans le processus top-down le concepteur analyse la requête reçue en s'appuyant sur les connaissances du domaine d'application. Par la suite, il procède à la collection des tâches qui satisfont la requête et leur organisation dans un workflow. Il est clair ici que la connaissance du domaine est un élément clé dans la conception top-down. Par conséquent, pour automatiser cela, il doit y avoir un modèle de représentation des connaissances, c'est à dire une ontologie, pour capturer la connaissance du domaine. Les travaux appartenant au premier mécanisme comme [Menascé *et al.*, 2008] et [Zhovtobryukh, 2007] s'appuient sur la connaissance contenue dans une ontologie de domaine pour générer un modèle de workflow abstrait. Dans le processus bottom-up, le concepteur connaît l'ensemble des services disponibles, ainsi que leurs spécifications détaillées. Suite à une requête, le concepteur sélectionne et relie les services appropriés pour obtenir une chaîne de service écrite, par exemple, en WS-BPEL. La liaison entre les services dépend du flux de données, impliquant le matching de jonction entre les services. Par ce mécanisme, le workflow est construit et il est appelé workflow dirigé par les données (*data-driven workflow*), car il est construit par les connexions du flux de données. Une définition formelle du workflow dirigé par les données est donnée par [Belhajjame *et al.*, 2008]. Dans le but d'automatiser le processus bottom-up, plusieurs travaux adoptant ce mécanisme ont exploité des techniques de planification de l'intelligence artificielle, en utilisant ce qu'on appelle le chaînage en avant et en arrière (*forward or backward chaining*) [Ren *et al.*, 2011] [Lécué et Léger, 2006]. Un cas particulier, le travail présenté dans [Zhovtobryukh, 2007], a proposé une approche théorique dirigée par les buts (*goal-driven Approach*) en adoptant le mécanisme top-down avec l'assistance de la méthode bottom-up. Dans ce travail, le but de la demande de composition est bien formulé et défini à l'aide d'un langage formel de description de but. Dans le travail présenté dans [Zhovtobryukh, 2007], un workflow abstrait est généré par la décomposition du but (top-down). La décomposition peut être basée sur une ontologie de domaine si elle est complète et décrit suffisamment le domaine respectif. L'approche de composition proposée par [Geyik *et al.*, 2013] combine également l'utilisation des méthodes top-down et bottom-up dans le but de produire de meilleurs résultats.

1.3.5.4 Sélection de services

Étant donné un modèle de workflow composé d'un ensemble d'activités abstraites, la sélection de services renvoie à la question : comment sélectionner efficacement un service approprié pour chaque activité. En effet, chaque activité peut avoir plusieurs services candidats identiques de point de vue fonctionnalité et qui ont différentes caractéristiques non fonctionnelles. Le but de la phase de sélection est de concrétiser un workflow en un workflow exécutable qui satisfait les critères non-fonctionnels. La figure 1.4 présente une illustration graphique. Dans cette figure, chaque cercle creux représente un service abstrait (activité), qui est une unité fonctionnelle non-exécutable et doit être instanciée par l'un de ses services concrets candidats correspondants qui sont représentés par des cercles pleins [Canfora *et al.*, 2005b].

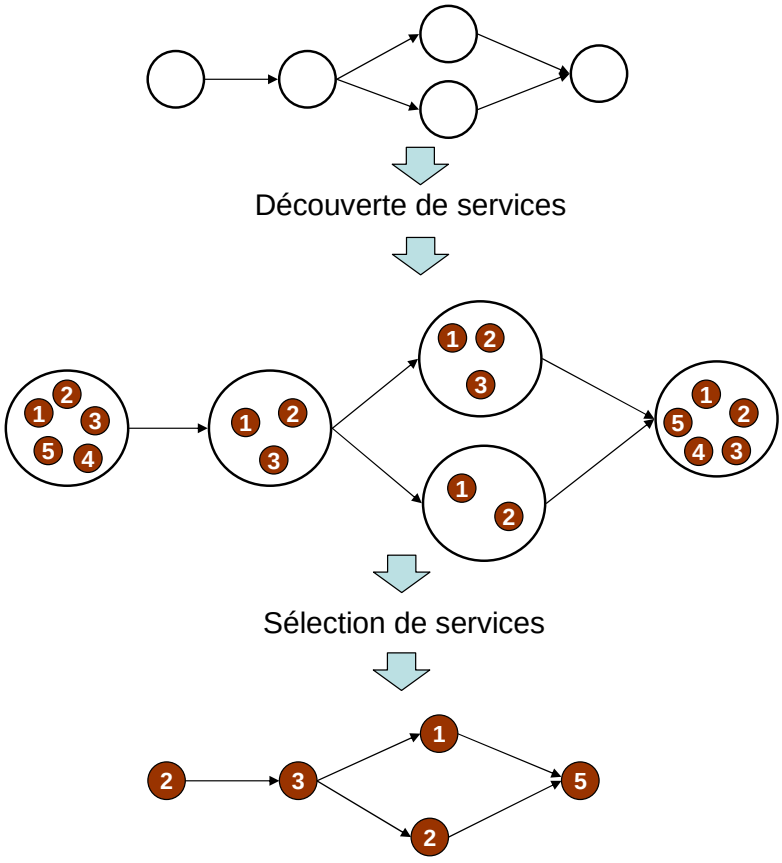

Figure 1.4 — Sélection de services

Initialement, un workflow prédéfini est constitué d'un ensemble d'activités abstraites, et chaque activité peut être entraînée par l'un des services concrets candidats après l'étape de découverte de service pour chaque activité. Les services concrets dans chaque groupe can-

didat d'une activité offrent typiquement des caractéristiques non fonctionnelles distinctes. Ainsi il s'agit, pour chaque activité, de sélectionner le meilleur service concret de chaque groupe candidat, générant ainsi un véritable service composite exécutable avec les propriétés non fonctionnelles nécessaires et acceptables. La sélection de services est la deuxième phase de la conception top-down, après la première phase de la création du workflow approprié. La sélection de services pourrait avoir lieu au moment de conception du service composite, au moment de l'exécution, ou les deux à la fois. Différentes possibilités peuvent se présenter. La première possibilité peut être comme suit : sélectionner les services appropriés au moment de conception, et puis effectuer le rebinding (binding dynamique) des services échoués au cours de l'exécution. La deuxième possibilité consiste à effectuer le binding total au moment de l'exécution, sans avoir sélectionné aucun service au moment de la conception. Nous allons ici nous concentrer sur la sélection de services qui se fait pour chaque activité contenue dans le workflow, plutôt que la re-sélection uniquement pour une ou deux activités. Il y a plusieurs aspects non-fonctionnels à prendre en considération lors de la phase de sélection, comme la qualité de Service (QoS) [Ramacher et Mönch, 2015], les propriétés transactionnelles [Haddad *et al.*, 2010] [FanJiang et Syu, 2014], les propriétés temporelles des processus métier [Liang *et al.*, 2013] [Guidara *et al.*, 2014], la fiabilité [Wang *et al.*, 2007], la prédiction de la mobilité [Wang, 2011], et la charge de la machine (consommation de ressources). Dans le contexte de la composition automatique, la sélection tenant compte de la QoS a été une problématique de recherche importante pour des années. Beaucoup de travaux ont été dédiés pour ce type de sélection, par exemple [Yu *et al.*, 2007], [Liang *et al.*, 2009], [Fujii et Suda, 2009], [Yau et Yin, 2011], [Lin *et al.*, 2012] et [Ramacher et Mönch, 2015]. Les caractéristiques de QoS assurées par le service composite ou exigées par le demandeur de service peuvent être une seule valeur représentative, qui est calculée par l'intermédiaire de certaines fonctions d'agrégation de QoS. Généralement, il existe deux différentes politiques pour la sélection QoS-aware : optimisation locale et globale. L'optimisation globale est préférable et adoptée par la plupart des travaux, car elle est plus précise que l'optimisation locale. Les types d'approches de la sélection peuvent être plus ou moins divisés en trois : la programmation linéaire, les algorithmes génétiques, et les algorithmes heuristiques.

1.4 Approches automatiques pour la composition des services

Comme nous l'avons mentionné précédemment, la composition automatique de services est un sujet académique qui couvre un champ important, impliquant et profitant de plusieurs domaines et techniques de l'informatique (par exemple le web sémantique, l'intelli-

gence artificielle, les ontologies...). Dans cette section nous présentons les principales catégories des approches de composition automatiques à savoir les techniques de planification de l'intelligence artificielle, les techniques basées sur le chaînage, les approches basées sur les règles et celles basées sur les connaissances d'une manière générale.

1.4.1 Techniques basées sur la planification IA

Plusieurs travaux de recherche ont exploité les techniques de planification de l'intelligence artificielle pour résoudre le problème de la composition de services comme [Peer, 2004], [Sirin *et al.*, 2004] et [Tang *et al.*, 2013]. Un problème de planification de l'IA traditionnelle est défini par un état initial, un état cible qui représente l'objectif du plan à générer et un ensemble d'actions. L'objectif de la planification en IA est de trouver un chemin à partir de l'état initial jusqu'à l'état cible. Ce chemin constitue le plan d'actions, à savoir une séquence d'actions. En général, un service composite correspond à un système états-transitions, qui a des états multiples et représente des transitions à partir d'un état initial acceptant les inputs d'un utilisateur jusqu'à un état final fournissant les outputs et les effets requis. Dans le système de transition, les actions applicables correspondent à des services disponibles. Si les préconditions et les inputs nécessaires d'une action sont satisfaits dans un certain état, la transition de cet état à un autre état peut être effectuée par l'application de l'action. Du coup, beaucoup de travaux autour de la composition de services automatisée essayent de résoudre le problème de la composition en le convertissant en un problème qui consiste à trouver les systèmes de transition adéquats. Pour cette raison, différentes techniques de l'IA tel que la planification HTN (*Hyper Task network*), la planification de contingence (*contingency planning*), la programmation par contraintes, et les preuves de théorème de la logique linéaire sont utilisés.

La planification basée sur le réseau hiérarchique des tâches (*HTN*) [Erol *et al.*, 1995] est une méthode de planification par la décomposition des tâches. Contrairement aux autres concepts de planification, le concept central de HTN n'est pas États, mais des tâches. Un système de planification basé HTN décompose de manière itérative la tâche désirée en un ensemble de sous-tâches jusqu'à ce que l'ensemble de tâches résultant soit constitué uniquement de tâches atomiques ou primitives, qui peuvent être exécutées directement par l'invocation de certaines opérations atomiques. Une approche utilisant la planification HTN dans le domaine des services web a été proposée dans [Evren *et al.*, 2003]. Le système de planification SHOP2 utilise les descriptions DAML-S pour la composition automatique de services. Sirin et al. [Sirin *et al.*, 2004] proposent une méthode de composition de services

web qui considère la sémantique fonctionnelle des services et améliore la complexité du temps du processus de composition basé sur la planification HTN. En se basant également sur la planification HTN, [Lin *et al.*, 2008] présentent une approche de composition automatique de services Web répondant aux préférences de l'utilisateur autant que possible. Les utilisateurs expriment des contraintes sur les États et précisent la trajectoire d'états correspondant au plan en utilisant un langage particulier. Le travail décrit principalement la façon dont les préférences sont traduites en un langage de planification pour les HTNs. Dans [Xiao *et al.*, 2010], un algorithme de planification HTN amélioré a été introduit pour réduire la profondeur et l'étendue du processus de planification. Les auteurs dans [Tang *et al.*, 2013] proposent un framework pour une composition de services web dynamique basé HTN. Ils développent un algorithme de matching de services à deux étapes qui permet de réduire le coût du temps du processus de composition.

Les auteurs dans [Chen *et al.*, 2009] considèrent que les approches de composition précédentes, fondées sur la planification HTN, ne prennent pas en considération le choix des décompositions disponibles ce qui peut conduire à une variété de solutions valables. Les auteurs présentent un modèle qui combine le modèle de décision du processus de Markov et la planification HTN pour aborder la composition de services Web. Dans ce modèle, la planification HTN est renforcée par la décomposition d'une tâche de façons multiples et donc être en mesure de trouver plus qu'un seul plan, en tenant compte à la fois des propriétés fonctionnelles et non fonctionnelles. En général la méthode de planification HTN prend en compte un but de l'utilisateur à décomposer. Néanmoins, la requête d'un utilisateur peut comporter plusieurs buts indépendants qu'il faut combiner au moment de la réception de la requête.

Rao et al. [Rao et Küngas, 2004] convertissent les spécifications de services et les besoins des utilisateurs en des axiomes et des théorèmes de la logique linéaire, respectivement, et puis ils essayent de trouver le service composite approprié par la démonstration de théorèmes (*theorem proving*). Agarwal et al. [Agarwal *et al.*, 2005] classifient les services web selon leurs interfaces et créent un service composite par composition logique et physique. Dans l'étape de composition logique, ils construisent un service composite, comprenant des branches conditionnelles, par l'application des plans de contingence. Kona et al. [Kona *et al.*, 2007] utilisent la programmation par contraintes pour découvrir un service web et établir un service composite. Leur méthode vérifie si les outputs et les effets exigés sont accessibles à partir des inputs et les préconditions fournis en utilisant les services disponibles. Ensuite ils construisent le service composite approprié en se basant sur cette « accessibilité ». Le travail présenté dans [Na-Lumpoon *et al.*, 2014] propose un framework qui fournit une compo-

sition automatique de services basée sur le calcul facile (*Fluent Calculus*). Cette étape, qui utilise la technique de l'inférence de la planification IA en association avec la méthode de programmation par contraintes de flux, génère un plan de composition qui contient des actions en séquence ou en parallèle. L'approche proposée dans [Hatzi *et al.*, 2012] est basée sur la transformation du problème de la composition de services web en un problème de planification et sa résolution après son enrichissement avec les informations sémantiques extraites de OWL-S.

Cependant, les méthodes mentionnées ci-dessus supposent que chaque service a des préconditions et des effets. Par conséquent, ces méthodes seraient incapables de construire un service composite en utilisant des services disponibles ayant uniquement des paramètres I/O. D'une autre part, il peut y avoir de multiples services avec des fonctionnalités différentes, bien qu'ils ont le même type de paramètres d'entrée et de sortie. La composition résultante peut ne pas satisfaire l'intention de l'utilisateur. Par ailleurs, étant donné que toutes les combinaisons possibles des services disponibles doivent être prises en compte dans les processus de composition, la complexité de temps est assez importante.

Typiquement, une planification en IA produit un service composite constitué par des actions atomiques sans hiérarchie sans tenir compte de l'information contextuelle. Srivastava et Koehler, dans [Srivastava et Koehler, 2003], identifient quelques situations particulières du problème de la composition de services Web qui rendent difficile l'application directe des algorithmes de planification IA, parmi lesquelles :

— Les plans nécessitent des structures de contrôle complexes impliquant des boucles, du non-déterminisme, et des structures conditionnelles.
— La planification doit prendre en compte la possibilité que non pas tous les artefacts de processus sont disponibles dès le début, certains d'entre eux sont des objets intermédiaires générés par les services Web au moment de l'exécution.

1.4.2 Techniques basées sur le chaînage

La technique basée sur des algorithmes de chaînage cherche à trouver les dépendances entre les différents services afin de synthétiser un plan de composition qui satisfait la requête. Dans cette catégorie, [Arpinar *et al.*, 2005] proposent la recherche d'un chemin de composition basé sur l'application de l'algorithme du plus court chemin sur le graphe des services. Dans ce travail, le graphe est exploré à l'aide d'un algorithme de chaînage en avant (*forward-chaining*) deux fois : la première passe permet de déduire tous les plans de composition possibles, et la seconde passe permet de choisir le chemin optimal. Sachant que

l'objectif de cette approche consiste à satisfaire les outputs de la requête, l'algorithme du chaînage en avant n'est pas suffisant et l'exploration en deux passes consomme beaucoup de temps. Dans [Aversano *et al.*, 2004], les auteurs utilisent chaînage en arrière (*backward chaining*) pour explorer le registre UDDI universel, mais le temps de réponse est trop long parce qu'il peut y avoir un très grand nombre de services qui sont indépendants. [Hashemian et Mavaddat, 2006] stockent les dépendances I/O entre les services disponibles dans un graphe de dépendance, ensuite ils construisent les services composites en appliquant un algorithme de recherche sur le graphe. Dans ce graphe, chaque service est représenté comme un sommet. L'entrée et la sortie d'un service sont représentés comme des arcs entrants et des arcs sortants, respectivement. Grâce à leur graphe de dépendance, Hashemian et Mavaddat peuvent seulement rechercher des services connectables, ce qui rend possible d'avoir une faible complexité de temps. Cependant, ils ne peuvent pas garantir que les services composites générés fournissent correctement la fonctionnalité demandée, puisqu'ils ne considèrent que le matching et les dépendances entre les paramètres d'entrée et de sortie sans tenir compte de la sémantique fonctionnelle de chaque service. Les auteurs de [Mohr *et al.*, 2015] proposent également l'utilisation d'un algorithme de recherche en arrière dans le but de traverser l'ensemble de toutes les compositions possibles. Les services disponibles sont déterminés au cours du processus de recherche.

Dans le travail de [Chan et Lyu, 2008], un algorithme de composition de service Web dynamique est présenté avec une vérification basée réseaux de Petri. Chaque service Web est décrit par WSDL et les interactions avec d'autres services sont décrits par WSCI. L'algorithme compose les services Web avec des informations fournies par ces deux descriptions. Après la composition, l'utilisation d'un réseau de Petri est nécessaire pour la vérification de l'absence de deadlock. La transformation vers un réseau de Petri utilise BEPL. Pour les auteurs, l'assurance d'avoir une composition sans deadlock justifie le temps dépensé pour la transformation vers un réseau de Petri et la vérification.

1.4.3 Approches basées règles

En ce qui concerne les approches à base de règles, un système fondé sur des règles est une combinaison d'un certain nombre de règles et un ensemble de conditions d'activation de ces règles. Elles sont simples dans le sens où elles offrent une fonctionnalité similaire aux déclarations *if-then-else*, mais dans une manière plus élaborée. Elles constituent une base de connaissances qui guide l'activation d'un comportement en se basant sur la connaissance du système offerte par les règles et les modèles d'activation. Une approche qui intègre la

programmation orientée objet et le raisonnement à base de règles a été fourni par D'Hondt et Jonckers [D'Hondt et Jonckers, 2004].

Les approches basées sur les règles définissent des règles spécifiques qui guident le processus de composition de services. Dans [Medjahed *et al.*, 2003] un modèle de composabilité qui vérifie quels services web peuvent interagir les uns avec les autres est décrit. Ces règles de composabilité vérifient les propriétés syntaxiques et sémantiques des services. Sur la base de ce modèle de compasibilité, une approche pour la composition automatique de service constituée de quatre phases est proposée : la phase de spécification permet une description haut niveau des services composites via le langage de spécification de service composite (CSSL), la phase de matching utilise les règles de composabilité pour générer des plans de composition, la phase de sélection sélectionne le meilleur plan et la phase de génération fournit une description du service composite dans un langage de composition choisi. Dans [Chun *et al.*, 2005], des règles de compatibilité sont introduites dans la composition de services. Les services sont composés sur la base de la compatibilité entre les politiques de service imposées par les fournisseurs, les politiques du flux de services et les politiques de l'utilisateur qui représentent les besoins des utilisateurs. Des règles de compatibilité syntaxique et sémantique sont également utilisées.

Mokhtar et al. [Mokhtar *et al.*, 2005] ont proposé un système de composition de services sensible au contexte qui étend OWL-S pour intégrer des règles telles que « la distance à l'utilisateur < 300m » ou « mémoire restante > 128Ko » et composer des applications basées sur ces règles intégrées. Bien que cette approche permet au concepteur de spécifier explicitement comment composer des applications pour un contexte donné, elle ne peut pas s'adapter à des utilisateurs différents parce que (1) des règles prédéfinies ne peuvent généralement pas être modifiés une fois qu'elles sont déployées, (2) il est difficile de définir une règle générique qui est applicable à tous les utilisateurs, et (3) la préférence de certains utilisateurs peut être trop complexe à définir comme étant un ensemble de règles.

Les auteurs de [Zeng *et al.*, 2008] proposent un mécanisme d'inférence à partir de règles pour générer dynamiquement un schéma de composition. Cette approche est basée sur un ensemble de règles métier spécifiques au domaine enrichies avec les informations contextuelles. Malgré l'idée intéressante de combiner l'utilisation des règles d'inférence de chainage-avant et de chainage-arrière, tout comme le travail cité précédemment, les règles métier sont implicitement codifiées dans le schéma de composition (en termes de contraintes sur les flux de contrôle et de données).

1.4.4 Approches basées connaissances

La composition à base de connaissances implique les connaissances spécifiques à un domaine dans le processus de composition de services. La plupart des approches de composition à base de connaissances se basent sur l'utilisation des ontologies. Dans [Chen *et al.*, 2003b], les connaissances spécifiques à un domaine sont utilisées pour guider la composition de services. Ce travail utilise les ontologies comme modèle de connaissances avec un vocabulaire commun. Le framework de composition proposé exploite les connaissances spécifiques au domaine et fournit des recommandations pour composer les services. Le travail présenté dans [Xu *et al.*, 2010] propose d'utiliser le modèle ER (entity-relationship), qui est largement utilisé dans la gestion de base de données, pour construire l'ontologie de domaine et annoter et composer les services. Dans le travail de [D'Mello et Ananthanarayana, 2010], un courtier pour la composition dynamique de services Web est proposé. Le courtier maintient connaissances de la composition et stocke la dépendance entre les opérations de service web et leurs paramètres de sortie et d'entrée. Les mécanismes de découverte et de composition des services se basent sur les concepts de la sémantique fonctionnelle et de la sémantique des flux des opérations de service Web.

1.4.4.1 Approches basées patrons

Les patrons peuvent être définis comme des ensembles de règles génériques qui peuvent être utilisés pour générer une solution à un problème. Dans le travail présenté dans [Tut et Edmond, 2002], l'utilisation des patrons dans la composition de services est introduite. En supposant que les patrons peuvent être utilisés pour représenter la logique des processus métier réutilisable, l'utilisation des patrons, notamment pour la représentation des exigences non-fonctionnelles, est illustrée à travers un exemple de mécanisme de paiement. L'utilisation des patrons pour la coordination des services de différents intervenants est proposée dans le travail de [Zirpins *et al.*, 2004]. Comme les intervenants changent de façon dynamique lors de la réalisation des applications à base de services, la logique d'interaction abstraite des services se concrétise seulement au moment de traitement de la requête. Différentes alternatives de coordination peuvent être réalisées à ce point. Ce travail décrit l'utilisation des patrons génériques d'interaction au moment de conception qui servent comme base pour l'analyse et l'optimisation de la logique d'interaction au moment de l'exécution. Une autre catégorie de patrons, à savoir les patrons d'usage, ont été utilisés dans le travail de [Chen, 2005] pour la composition des services. Dans ce travail, un système de filtrage collaboratif est utilisé pour le traitement des patrons d'usage comprenant les décisions passées

des utilisateurs qui ont été enregistrées avec des informations du contexte de l'utilisateur. Sur la base de cette information, il est possible de prédire la façon dont les utilisateurs avec des profils similaires se comportent dans la même situation. Il serait possible donc d'utiliser des analyses statistiques sur un grand nombre de ces patrons d'usage de services pour obtenir automatiquement des ensembles de services qui seront utilisés par des souscripteurs avec des profils similaires et dans des contextes similaires. Ces services sont, par définition, les candidats idéaux pour la composition d'un nouveau service qui accomplit la tâche souhaitée.

1.4.4.2 Approches basées sur le raisonnement par cas (et/ou) apprentissage

Les auteurs dans [Lajmi *et al.*, 2006] proposent une approche appelée WeSCo_CBR (*web Service Composition founded on Case Based Reasoning*) pour la composition des services web. Elle est basée essentiellement sur le raisonnement à partir de cas. Ils proposent ensuite d'améliorer le processus de recherche des cas similaires par la classification des cas. Le raisonnement à partir de cas est le processus de résolution de nouveaux problèmes en se basant sur les solutions de problèmes similaires passés. En d'autres termes, au lieu de se baser uniquement sur les connaissances générales du domaine d'un problème, ou faire des associations le long des relations généralisées entre les descripteurs du problème et les conclusions, le raisonnement à partir de cas est capable d'utiliser les connaissances spécifiques des cas des problèmes concrets déjà rencontrés et résolus. Dans ce travail, la méthode CBR est appliquée, combinée avec l'utilisation du langage OWL-S pour décrire et développer les processus abstraits. Ce type de raisonnement consiste à trouver, dans la base de cas, des cas similaires à une nouvelle requête d'un utilisateur. Certains systèmes de composition sensibles au contexte utilisent des techniques d'apprentissage pour identifier les services qui doivent être sélectionnés dans un contexte donné en se basant sur l'historique des compositions passées. Par exemple, [Casati *et al.*, 2004] a proposé un système de composition de service sensible au contexte qui construit des arbres de décision basés sur des contextes passés et qui utilise ces arbres pour sélectionner les services au moment de l'exécution. Cette approche basée sur l'apprentissage peut s'adapter aux différents utilisateurs. Les auteurs dans [Wang *et al.*, 2014] traitent la problématique de la composition à partir d'un grand nombre de services dans un environnement dynamique. Ils proposent un modèle pour la composition et l'adaptation qui intègre les connaissances de l'apprentissage par renforcement. En particulier, un algorithme basé *Q-learning* pour la composition de services sur la base du modèle proposé est également défini.

Toutefois, selon [Fujii et Suda, 2009], les systèmes à base d'apprentissage existants ne sont pas aussi efficaces que les systèmes basés sur les règles car ils ont besoin d'accumuler une certaine quantité des informations historiques à chaque fois qu'un nouvel utilisateur entre ou un utilisateur demande une nouvelle application composée de nouveaux services avant d'effectuer correctement l'apprentissage des préférences de l'utilisateur.

1.5 Problématique

Les travaux qui traitent la composition de services touchent plusieurs aspects notamment la description de services, l'automatisation de composition, l'utilisation de la sémantique et la généricité de la composition.

Nous pouvons dire que l'état de l'art concernant la composition automatique des services met en évidence quelques limitations et un certain nombre de points problématiques. Nous exposons ces points dans ce qui suit.

1.5.1 Besoin de flexibilité dans la construction de services composites adaptables

La plupart des approches de composition existantes supposent que les schémas de composition sont statiques et prédéfinis. Ce mode de composition nécessite que toutes les relations entre les taches soient établies à priori. Il exige par ailleurs la transformation minutieuse des règles métier et des relations en un modèle de processus particulier. Dans le cas où plusieurs alternatives existent pour atteindre un objectif, ces alternatives doivent être énumérées et examinées de sorte que la stratégie optimale peut être sélectionnée. Ceci peut conduire à un grand nombre de chemins d'exécution étant définis et une incapacité à faire évoluer le processus au moment de l'exécution. En outre, il n'est pas toujours possible de prédéfinir les schémas de processus pour des domaines d'application sophistiqués et volatils. Par conséquent, il existe un besoin pour la génération dynamique et automatique des schémas de composition adaptés aux besoins de l'utilisateur.

Dans notre approche, nous introduisons de la flexibilité en permettant le choix des services abstraits en runtime, en s'appuyant sur le contexte. La description de services est une activité qui peut se faire à travers plusieurs langages, notamment pour les services web. Si le concept de service est considéré avec plus d'abstraction, il est possible d'introduire plus de flexibilité, car des implémentations différentes pourraient le réaliser. Il parait donc nécessaire d'introduire une description pour les services abstraits.

1.5.2 Difficultés pour la génération dynamique des schémas de composition

Les approches de composition dynamiques visent à introduire la flexibilité dans la construction de services composites sans avoir recours à des plans de composition prédéfinis. Typiquement, les techniques de composition issus de la planification de l'intelligence artificielle produisent dynamiquement un service composite constitué d'un ensemble d'actions atomiques sans considération des informations contextuelles. Un autre aspect qui rend difficile l'application directe des algorithmes de planification est que les plans de composition nécessitent des structures de contrôle complexes impliquant des boucles, du non-déterminisme, et le choix. Dans notre approche, nous nous attaquons à ce problème en encapsulant les structures de contrôle complexes dans les services abstraits définis au moment de la conception. Au moment de l'exécution, nous nous concentrons sur le chaînage des services.

Dans notre travail, nous introduisons un modèle pour les services abstraits qui peut être utilisé à travers les outils de la sémantique. La sémantique permet de définir les règles de raisonnement et des techniques de validation et de vérification qui servent la composition de services. L'automatisation de la composition fait l'objet de nombreux travaux de recherche qui essaient d'automatiser totalement la composition. Cependant, l'intervention de l'humain reste nécessaire. A travers la sémantique nous assurons une intervention minimale de l'humain.

1.5.3 Inadéquation de la composition de service avec l'intention de l'utilisateur

Plusieurs approches existantes qui s'intéressent à la composition automatique de services comme le travail présenté dans [Kona *et al.*, 2007] utilisent une méthode de composition qui vérifie si les outputs et les effets exigés sont accessibles à partir des inputs et les préconditions fournis en utilisant les services disponibles. Ensuite, ils construisent le service composite approprié en se basant sur cette accessibilité. Ces approches supposent que chaque service a des pré-conditions et des effets. Si les services existants n'ont pas de pré-conditions ou des effets, leur méthode doit construire un service composite en utilisant seulement les paramètres d'entrée/sortie correspondants. Le problème est qu'il peut y avoir de nombreux services avec des fonctionnalités tout à fait différentes, même si elles ont les mêmes paramètres d'entrée et de sortie. Dans ce cas, le service composite résultant peut ne pas satisfaire les intentions de l'utilisateur.

Dans notre travail, nous définissons explicitement la sémantique des services abstraits, utilisés pour établir la composition, afin de sélectionner ceux qui répondent convenablement

aux intentions de l'utilisateur. Nous constatons que la plupart des travaux se focalisent soit sur la façon optimale de sélectionner des services correspondants aux besoins des utilisateurs, soit sur la combinaison de services en omettant parfois l'adéquation de la réponse aux besoins des utilisateurs. Dans ce travail, nous tenons compte des deux aspects. Nous proposons une approche qui se veut la plus générique possible en traitant la composition d'une manière holistique ce qui permet de réduire le fossé entre l'expression du besoin de l'utilisateur et le service concret, et de générer le schéma de composition optimal.

1.6 Conclusion

Dans ce chapitre, Nous avons présenté une étude au tour des concepts de service et de composition de service. Nous avons présenté les concepts de base liés aux services et les architectures à services. Ensuite, nous nous sommes focalisés sur la composition de services et les principaux défis soulevés dans ce domaine. Notre intérêt s'est porté sur les approches et techniques de composition automatique de services. Nous avons abouti à la problématique pour mettre en évidence le besoin de flexibilité dans la construction de services composites adaptables, les difficultés pour la génération dynamique des schémas de composition et les problèmes qui résultent de l'inadéquation de la composition de service avec l'intention de l'utilisateur.

2 Modèles sémantiques pour la composition des services

2.1 Introduction

Ce chapitre présente les modèles sémantiques que nous avons élaborés, utilisés pour effectuer la composition de services selon notre approche. Nous commençons d'abord par introduire intuitivement l'approche générale de composition proposée dans le cadre de cette thèse. Le processus de composition est guidé par le modèle des intentions de l'utilisateur et basé sur l'utilisation des services abstraits. Ensuite, nous détaillons le modèle sémantique des services abstraits dont le but est de servir d'éléments définis au moment de la conception pour permettre une flexibilité dans la composition au moment de l'exécution. Ce modèle est basé sur l'utilisation du langage OWL-S que nous allons introduire brièvement. Par la suite, nous présentons le modèle des intentions qui va servir de base pour la génération du schéma de composition.

2.2 Approche générale

Nous proposons une approche pour une composition de services flexible au moment de l'exécution pour la sélection des services adaptés à un utilisateur. Notre approche consiste à générer automatiquement un schéma de composition qui définit un flux d'activités et leur flux de contrôle sans identifier les services concrets à invoquer. Il s'agit ensuite de sélectionner et assigner les services concrets appropriés aux services abstraits du schéma de composition pour former le plan d'exécution. Ce processus de composition qui est effectué au moment de l'exécution est guidé par une spécification des intentions de l'utilisateur. Il est basé par ailleurs sur l'utilisation d'un ensemble de services abstraits préexistants. L'approche proposée tire profit des avantages de la composition dynamique pour assurer un certain niveau de flexibilité et parsuite d'adaptabilité, et de l'utilisation de fragments de processus

composés (ou services composites) prédéfinis pour faciliter la composition automatique.

Les services abstraits sont les composants fournis au moment de la conception de notre solution. Ils sont utilisés pour réduire la difficulté des tâches de découverte et de sélection des services concrets qui couvrent les exigences des utilisateurs. Les services abstraits représentent des descriptions génériques et réutilisables. Cela permet la spécification de processus génériques pour différentes situations. Il est soutenu que la plupart des tâches requises sont répétées régulièrement, et les étapes de base pour la réalisation de ces tâches sont connues, au moins à un niveau abstrait. Néanmoins, la réalisation de ces tâches varie car elle est adaptée aux utilisateurs. À ce titre, notre proposition est de spécifier ces tâches à l'aide d'un ensemble de services abstraits présentant éventuellement des services alternatifs, et de les combiner et les raffiner au moment de l'exécution.

Les étapes impliquées dans le mécanisme de composition que nous avons proposé sont présentées dans la figure 2.1. L'entrée du processus de composition est une spécification des intentions de l'utilisateur. Cette spécification décrit une structuration des besoins de l'utilisateur. Dans notre travail, nous avons formalisé les intentions en tant que structures sémantiques qui représentent les besoins d'un utilisateur. L'intention est en relation avec un contexte. Ce contexte est représenté par une ontologie qui permet de structurer sémantiquememt les relations contextuelles et permet une réalisation des besoins de l'utilisateur en adéquation avec son environnement. A travers l'application d'un ensemble de règles que nous avons définies, la spécification des intentions est transformée en une spécification plus enrichie. Ensuite, cette spécification permet de générer un schéma de composition initial par l'application des règles de composition et la mise en correspondance des intentions aux services abstraits. La mise en correspondance se fait à travers des règles sémantiques exploitant une ontologie de domaine. Par ailleurs, les informations structurées fournies par la spécification des intentions permettent de combiner les services produisant un schéma de composition approprié appelé schéma de composition initial. Ce dernier est optimisé en étudiant le degré d'affinité entre les services qui le constitueraient. Ensuite le schéma de composition initial est raffiné, ayant recours également à une ontologie de domaine et utilisant l'ontologie du contexte pour faire appel à des services abstraits les plus adaptés aux situations contextuelles, jusqu'à obtenir un schéma de composition final composé de processus atomiques. Les services concrets qui peuvent exécuter ce schéma de composition sont sélectionnés en fonction des contraintes non-fonctionnelles exprimées dans les intentions. Ainsi, le plan d'exécution est généré avec les services concrets.

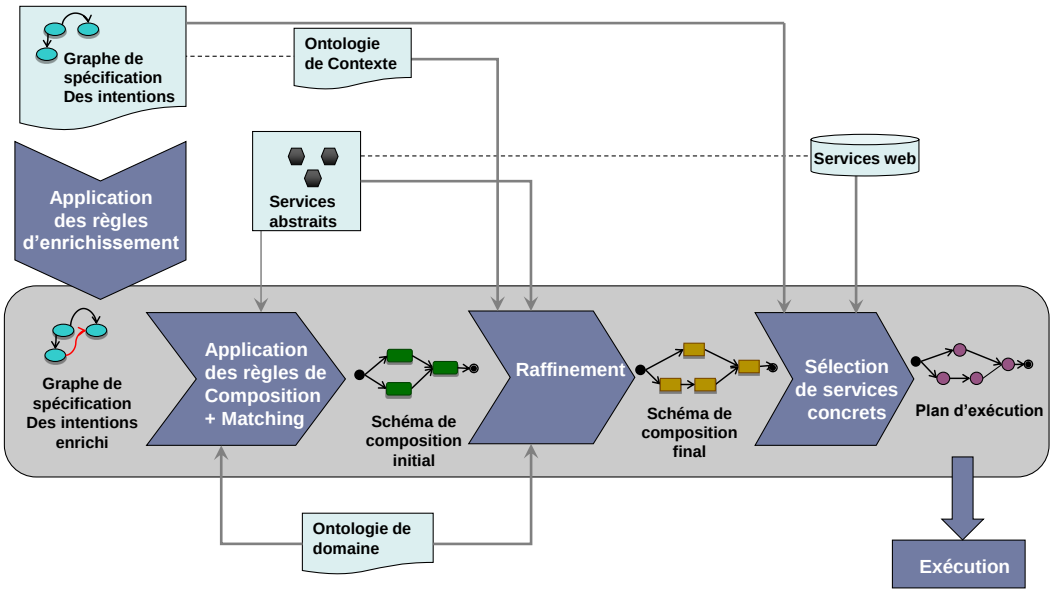

Figure 2.1 — Processus de composition de services

2.3 Pré-requis de notre approche de composition de services

Cette partie présente principalement nos modèles pour les intentions et les services abstraits qui sont nécessaires pour la construction d'une composition de services dans notre approche. Ces modèles sont décrits par des ontologies. Le but principal du modèle des intentions est de servir de point de référence pour l'expression des besoins de l'utilisateur. Ce modèle est indépendant du domaine considéré, mais il peut être associé à un modèle spécifique du contexte pour la prise en compte de domaines concrets. Le deuxième but du modèle des intentions est de servir de base pour la découverte des services impliqués dans la composition et la déduction du flux de contrôle à travers ces services. Nous avons opté pour le langage OWL [McGuinness et van Harmelen, 2004] pour la représentation des intentions des utilisateurs et l'ontologie de contexte.

Le modèle des services abstraits fournit une description des services ainsi qu'une structure de composition favorisant l'automatisation du processus de composition global. Ce modèle est basé essentiellement sur l'ontologie des services OWL-S qui est fondée sur le langage OWL.

Dans ce qui suit, nous introduisons les ontologies OWL utilisées pour décrire nos modèles. L'utilisation du langage OWL-S pour décrire nos services abstraits nous conduit également à donner quelques éléments de base de OWL-S.

2.3.1 Introduction aux ontologies OWL/OWL-S

OWL est un langage XML profitant de l'universalité syntaxique de XML. Il offre un moyen d'écrire des ontologies web. Le langage d'ontologie Web OWL est conçu pour décrire des classes et leurs relations, lesquelles sont inhérentes aux documents et applications Web. Un des points forts de OWL est sa capacité de décrire les classes d'une façon plus intéressante et plus complète que le RDF et le RDFS. En plus, grâce à son mécanisme d'importation, il permet d'incorporer plusieurs ontologies dans une nouvelle ontologie accélérant ainsi le processus de développement des ontologies. En conséquence, il est impératif d'utiliser le langage OWL pour pouvoir bénéficier de tous les avantages et de toute la puissance sémantique des ontologies.

Pour pouvoir faire des inférences sur les connaissances décrites en OWL, ce langage est étendu par un autre langage dérivé qui est SWRL. Ce dernier permet de définir des règles d'inférence pour interroger les connaissances exprimées en OWL dans une ontologie. Il permet également de produire de nouvelles connaissances inférées à partir de celles qui existent dans l'ontologie.

Profitant des avantages du langage OWL, le langage OWL-S a été défini pour décrire les services Web de façon non ambiguë et interprétable par des programmes. En effet, le OWL-S est basé sur le langage d'ontologie OWL. La description des services web avec OWL-S est motivée par un certain nombre d'objectifs : (1) la Découverte automatique de services Web, (2) l'invocation automatique des services Web et (3) la composition automatique des services Web. Pour atteindre ces objectifs, OWL-S définit une ontologie supérieure pour la description, l'invocation et la composition des services Web. La structuration de l'ontologie supérieure de OWL-S permet de distinguer trois éléments dans la description d'un service (figure 2.2), à savoir :

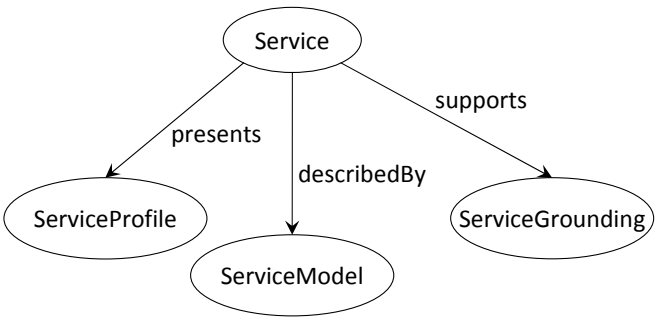

Figure 2.2 — Aperçu général de l'ontologie OWL-S

— Profil de service : cette information est donnée par la classe *ServiceProfile*. Ce dernier

permet la description et la découverte des services, en spécifiant une description des propriétés fonctionnelles et des propriétés non fonctionnelles. La partie *Profile* est utilisée à la fois par les fournisseurs pour publier leurs services et par les clients pour spécifier leurs besoins. Elle constitue par conséquent l'information utile pour la découverte et la composition de services.

— Modélisation de services : Les services peuvent etre modélisés avec OWL-S en tant que processus. La classe ainsi définie est *Process*, qui est une sous-classe de *Service-Model*. La partie *Process* est utilisée pour composer les services. OWL-S modélise les services en tant que processus. Chaque processus est défini par ses entrées/sorties. Il existe trois types de processus dans la partie relative à la modélisation de processus avec OWL-S : les processus atomiques (*AtomicProcess*), simples (*SimpleProcess*) et composites (*CompositeProcess*). Un processus atomique représente le niveau le plus fin pour un processus et ne peut pas être décomposé de façon plus profonde. Son exécution correspond à une unique avancée dans l'exécution du service. Les processus composites sont décomposables en d'autres processus ; leur décomposition peut être spécifiée en utilisant un ensemble de structures de contrôles tels que : *Sequence*, *Split*, *If-Then-Else*, etc. Un processus simple n'est pas invocable directement. Il fournit une façon simplifiée d'utiliser un service atomique, ou alors le moyen de voir sous une seule entité un processus composite.

— Détails d'accès à un service. ces détails sont fournis par le *Grounding* du service. Le *Grounding* indique comment accéder concrètement au service et fournit les détails concernant les protocoles, les formats de messages et les adresses physiques.

2.3.2 Les services abstraits

2.3.2.1 Définition et motivation

Nous définissons un service abstrait en tant que service qui a une description générique et réutilisable. L'idée derrière les services abstraits, est de faciliter la sélection et la composition des services. Ils peuvent aider à améliorer les performances et l'efficacité des compositeurs en diminuant le nombre de services. Un service abstrait peut être considéré comme un niveau d'abstraction intermédiaire dont le but est d'établir un pont entre les intentions des utilisateurs et les services concrets potentiels qui permettent de les réaliser. Cela réduirait la difficulté et le coût élevé de la tâche de la découverte et de sélection des services concrets qui couvrent les intentions des utilisateurs. Puisqu'un même service abstrait est offert à plusieurs circonstances correspondantes à différents utilisateurs, différentes instances

de ce service peuvent être obtenues à différents niveaux de granularité allant des propriétés fonctionnelles des services jusqu'aux propriétés liées à l'exécution. Les services abstraits proposés sont motivés par les principaux aspects suivants :

— Le besoin de faciliter la mise en correspondance entre les intentions des utilisateurs et les services concrets disponibles.
— Le besoin de flexibilité dans la composition de services pour satisfaire les intentions particulières.
— L'importance de la réutilisabilité de services.

Les services abstraits sont composés pour construire des processus complexes. A priori, dans notre travail, les services abstraits sont des entités prédéfinies fournies au moment de la conception par des experts de domaine. Néanmoins, ces services sont amenés à évoluer dynamiquement à travers des méthodes d'apprentissage ou de raisonnement par cas par exemple.

L'utilisation des services abstraits présente les avantages suivants :

— Les services abstraits peuvent être définis sans connaître l'implémentation des services web car l'implémentation ne modifie pas le service, même en cas de changement d'interface.
— Pendant son cycle de vie, un service abstrait peut être attaché à plusieurs services web.
— La distance sémantique entre les services abstraits et les intentions est plus stable que la distance entre les intentions et les services web. En effet, les services web changent plus fréquemment que les services abstraits. Ceci provoque un re-calcul de la distance plus fréquemment dans le cas de services web.
— Un service abstrait présente un fragment de processus qui peut être utilisé par plusieurs compositions de services. Ainsi, un service abstrait peut encapsuler des structures de contrôle complexes qui ne peuvent pas être générées automatiquement. Par conséquent, l'utilisation des services abstraits facilite la génération du schéma de composition au moment de l'exécution.

2.3.2.2 Spécification des services abstraits

Le service abstrait est décrit par son profil qui fournit les informations nécessaires pour décrire la fonctionnalité du service, notamment la fonctionnalité du service et l'ensemble des inputs et outputs. Comme [Ye et Bin, 2006], nous décrivons la fonctionnalité d'un service en tant qu'une action et un objet (l'objet de l'action). Par exemple, la fonctionnalité d'un

service qui permet la réservation d'un hôtel peut être spécifiée comme {réserver, hôtel}. Le service abstrait est composé d'une ou plusieurs activités. Un service abstrait est associé avec un type de service qui peut être atomique ou composite. Chaque service a une ou plusieurs activités qui sont utilisées pour satisfaire la fonctionnalité du service abstrait. La manière dont un service abstrait est constitué d'activités est en relation avec le type du service abstrait. Si le service abstrait est de type atomique, il est constitué d'une seule activité. Il est considéré comme un service opérationnel car il peut être directement implémenté par un service web concret. Si le service est composite, le service abstrait est constitué de plusieurs activités. L'exécution de ces activités peut être en séquence (ordre particulier) ou en parallèle (sans ordre précis). Par ailleurs, un service abstrait composite peut représenter dans certains cas un service conditionnel qui propose des alternatives pour satisfaire sa fonctionnalité. L'introduction de la variabilité dans la conception du service abstrait est justifiée par le besoin d'introduire de la flexibilité dans la réalisation des fonctionnalités et de l'adaptabilité dans le processus de composition de service. Au moment de la composition, une activité est sélectionnée en fonction des informations fournies par la spécification des intentions et les informations du contexte. Dans ce cas, des règles sont utilisées pour guider la sélection de l'activité adéquate.

2.3.2.3 Mise en œuvre des services abstraits avec OWL-S

Bien que OWL-S et la structure de son ontologie supportent les mécanismes d'abstraction pour représenter les services, OWL-S se focalise principalement sur la définition des services web concrets. Chaque service OWL-S est associé soit à un service Web physique ou à une composition fixe d'un ensemble de services Web concrets physiques, ce qui reste encore un seul service web concret vu de l'extérieur. Toutes les classes OWL-S, tels que *Profile*, *Process* et *Grounding*, contiennent différentes informations sur le service concret. En réalité, un service abstrait est aussi important qu'un service concret, et ceci est d'autant plus vrai si une ontologie de services est de taille importante. Un service abstrait généralement représente un ensemble de services concrets similaires, tandis qu'un service OWL-S concret correspond à un seul service Web concret.

Un service abstrait généralise les points communs d'un groupe de services concrets et fournit un certain niveau d'abstraction de ces services. Le langage OWL-S fournit des mécanismes d'abstraction pour représenter les processus. En outre, il fournit de l'information sémantique afin que les services puissent être découverts et interagissent d'une manière automatique. Pour ces raisons, nous avons exploité ce langage pour représenter et mettre en

œuvre nos services abstraits.

Nous considérons que le service abstrait est composé d'une partie *Profile* et une partie *Process*. La partie *Profile* décrit le nom, les entrées (inputs) et les sorties (outputs) du service abstrait. Dans le *Process* de OWL-S, un service est modélisé comme un processus qui est classé comme un processus atomique, un processus simple, ou un processus composite. Un processus simple pourrait être réalisé par un processus atomique ou étendu à un processus composite. La partie *Process* de notre service abstrait peut être mise en correspondance avec la partie *Process* de OWL-S. Cependant, ce *Process* ne peut être qu'un processus atomique, ou un processus composite, comme le montre la figure 2.3. Dans ce dernier cas, le processus du service abstrait est composé d'autres sous-processus qui sont des processus simples. Leur composition peut être spécifiée en utilisant l'une des structures de contrôle tels que *sequence* et *if-then-else*. Un processus simple peut servir comme une vue abstraite d'un processus atomique ou composite afin qu'on puisse voir le même processus à partir d'angles différents. Les processus simples ne sont pas invocables et ne sont pas associés à un *Grounding*. Le service abstrait est donc conceptuellement similaire à une classe abstraite dans les langages de programmation orienté objet. Dans le cas où le service abstrait a un processus atomique, le service est associé à un ou plusieurs services concrets OWL-S à travers la relation *implementableBy*. L'implémentation de chaque service concret OWL-S et les détails pour accéder à et invoquer le service réel sont spécifiés par la partie *Grounding*. Cet ensemble de services concrets OWL-S constitue une collection de services web ayant une fonctionnalité commune et présentant des propriétés non fonctionnelles différentes (par exemple : fournisseurs différents, valeurs de QoS différentes, etc). Ces propriétés non fonctionnelles sont spécifiées dans la partie profil de chaque service concret. Au moment de l'exécution, le service candidat approprié est sélectionné et invoqué. Pour les environnements et les frameworks qui exigent une adaptation ou une auto-réparation au moment de l'exécution, l'ensemble des services candidats peuvent préparer les services pour le remplacement ou la substitution.

2.3.3 Formalisation des intentions

Dans notre travail, nous avons formalisé les besoins des utilisateurs par des intentions qui permet de les structurer sémantiquement. Dans ce qui suit, nous présentons les travaux sur lesquels nous nous sommes appuyés pour utiliser les intentions. Ensuite, nous donnons notre représentation des intentions à travers un modèle sémantique.

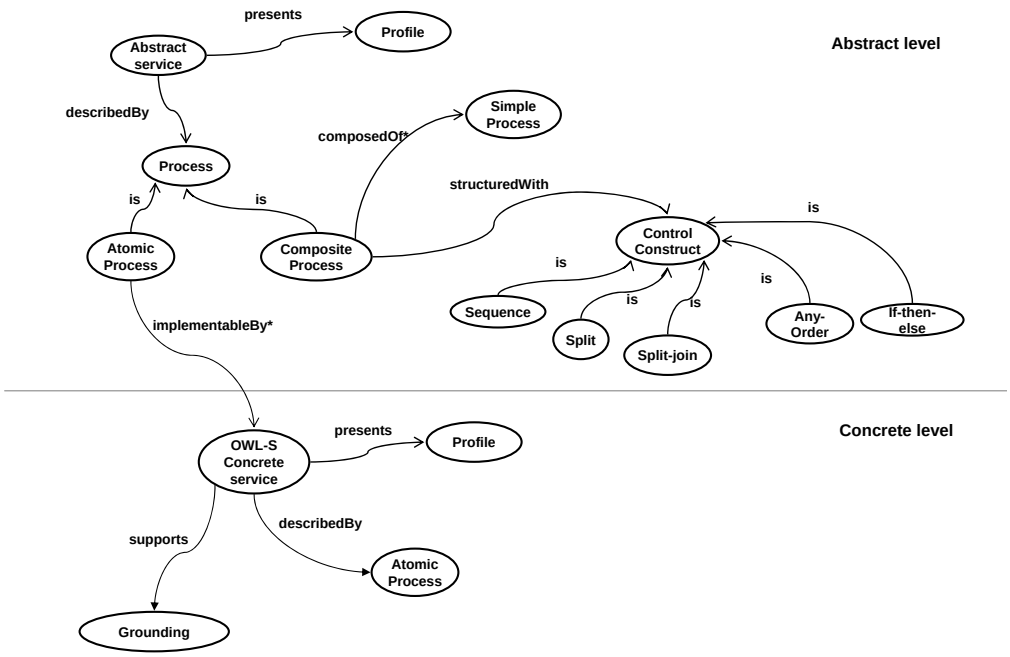

Figure 2.3 — Modèle du service abstrait

2.3.3.1 Background sur les intentions

Les intentions humaines ont été bien étudiées par de nombreux chercheurs en philosophie, en sciences cognitives [Tomasello *et al.*, 2005], et en Intelligence artificielle [Cohen et Levesque, 1990] [Rao et Georgeff, 1991]. La définition de l'intention reste toujours variée et parfois controversée. Les définitions ont souvent tendance à lier l'intention humaine aux états mentaux, et elles varient selon les domaines de recherche. Bratman, décrit l'intention comme l'un des états mentaux de BDI (Belief-Desire-Intention), qui motive l'action [Bratman, 1999]. Les chercheurs en IA ont utilisé leur propre modèle BDI et par conséquent, ont développé des techniques de planification basées sur les agents ; Cependant, dans la plupart des travaux basés sur l'IA, l'intention est considérée comme un plan d'exécution d'un agent [Morreale *et al.*, 2006]. Les buts sont parfois appelés des intentions puisqu'ils sont des concepts liés et complémentaires. À savoir, tout objectif d'une intention peut être représenté comme un but. Dans l'analyse des besoins pour les systèmes sensibles au contexte, les intentions sont capturées à partir de modèles comportementaux d'un utilisateur particulier. Par ailleurs, les buts sont objectifs, partageables entre les parties prenantes à travers la représentation explicite des connaissances [Ming *et al.*, 2008].

Pour notre travail, dans le contexte de découverte de services, nous considérons que les

intentions guident la manière dont les services sont fournis et composés. Pour l'exploitation des intentions de l'utilisateur, nous avons défini l'intention comme combinaison d'un but et un ensemble de moyens qui expriment comment ce but est accompli. Ces moyens sont composés d'un ensemble de contraintes fonctionnelles et non fonctionnelles. Nous considérons que l'extraction des intentions est une étape préalable à ce travail.

A notre connaissance, il n'y a pas de travaux qui se sont focalisés sur la composition des services basée sur les intentions de l'utilisateur. Toutefois, la notion de l'intention a été utilisée dans quelques travaux. Le travail de [Chang *et al.*, 2009] présente une approche de l'évolution de services dirigée par les intentions humaines dans les environnements de services sensibles au contexte. Dans ce travail, le framework présenté permet de modéliser et d'inférer les intentions humaines par la détection des désirs d'un individu ainsi que par la capture des valeurs de contexte correspondantes à travers des observations. Cela aide les développeurs à complèter la spécification des exigences pour les services. Dans notre travail, nous ne sommes pas intéressés par comment est capturée l'intention. Nous utilisons l'intention pour faciliter et optimiser la découverte et la composition de services.

Les auteurs de [Rolland *et al.*, 2010] proposent une évolution vers une description des services en termes d'intentions. Ils désignent ces services comme des services intentionnels. Ils proposent une méthodologie pour déterminer les services intentionnels qui répondent aux objectifs métiers. Ce travail met l'accent sur la capture des besoins en services par l'exploration des objectifs métier. Dans notre travail, nous nous concentrons sur la composition des services guidée par les intentions de l'utilisateur plutôt que la description des services avec les intentions. En effet, selon notre point de vue, les intentions sont spécifiques aux utilisateurs particuliers. Ainsi, nous considérons les services abstraits des composants génériques et réutilisables qui sont utilisés pour composer des services en se basant sur les intentions spécifiques des utilisateurs.

2.3.3.2 Représentation des intentions

Le modèle des intentions sur lequel se base le travail présenté dans cette thèse est principalement constitué d'un but et d'un ensemble de contraintes orientant la réalisation de ce but. Par exemple, pour un but tel que « *planifier une visite touristique* », nous pouvons avoir une contrainte fonctionnelle comme « *frais des visites <20 euros* ». En plus des contraintes fonctionnelles, un utilisateur de service peut avoir des contraintes non-fonctionnelles. Celles-ci peuvent être liées aux contraintes d'exécution (par exemple : le temps d'exécution d'un service). Elles peuvent être également liées à la sécurité. Par exemple, un service néces-

sitant une authentification peut être demandé par l'utilisateur. Une contrainte fonctionnelle de l'intention ayant pour but « *planifier une visite touristique* » est : « *visiter les sites dont le prix est inférieur à 20 euros* ». La contrainte non fonctionnelle est la suivante : « *les transactions doivent être sécurisées* ». Le but d'une intention est représenté par une paire : une action et l'objet de l'action. Par exemple, l'action du but « *Réserver hôtel* » est « *Réserver* » et son objet est « *hôtel* ».

Pour identifier les relations qui existent entre les intentions nous nous sommes basés sur la théorie de Grosz et Sidner [Grosz et Sidner, 1986]. Dans cette théorie, ces deux auteurs ont défini la structure intentionnelle qui permet de représenter la structure des objectifs (*purpose structure*) [Tazi, 2005]. Cette structure a été définie dans le cadre de l'élaboration des modèles de représentation des structures du discours dans les documents. L'objectif sous-jacent étant de permettre la reconnaissance par le lecteur des intentions de l'auteur. Dans cette théorie, les auteurs ont identifié deux relations structurelles entre intentions : la relation de dominance, et la relation de précédence de satisfaction :

— Une intention I_1 domine une intention I_2 si la satisfaction de I_2 contribue à celle de I_1.

— Une intention I_1 précède (la satisfaction de) I_2 si I_1 doit être satisfaite avant I_2.

Nous avons modélisé la spécification des intentions comme un graphe orienté attribué $GI =< I, R >$ où : I représente l'ensemble des intentions, et R représente les relations entre ces intentions. $I =< B, C >$ où : B représente le but de l'intention, et C représente l'ensemble des contraintes liées à la réalisation de l'intention. $C =< CF, CNF >$ où :

— $CF = \{cf_1, cf_2, ...\}$ est l'ensemble des contraintes fonctionnelles.

— $CNF = \{cnf_1, cnf_2, ...\}$ est l'ensemble des contraintes non fonctionnelles.

$R =< d, p >$; où d représente la relation de dominance et p représente la relation de précédence.

Les études sur la modélisation de l'intention dérivent principalement des théories causales de l'action [Tazi, 2005]. Décrire une intention revient à trouver une explication rationnelle à l'action qui a été causée par cette intention. Cette explication relève plus de la pragmatique situationnelle, c'est à dire : elle dépend du contexte dans lequel l'action peut se dérouler. Pour cette raison, il est évident que la modélisation des intentions ne peut se faire que par rapport aux contextes dans lesquels les actions ciblées par ces intentions peuvent se dérouler. Ainsi, nous considérons que chaque intention est associée à un ou plusieurs paramètres de contexte. L'existence du concept *ContextParameter* permet de lier l'ontologie des intentions à une ontologie de contexte. Cette ontologie peut contenir différents éléments de contexte et les règles sémantiques correspondantes. La figure 2.4 présente une ontologie

modélisant l'intention représentée par le concept *Intention*. Une ontologie peut être considérée comme un méta-modèle qui décrit les concepts et les relations possibles d'un domaine donné.

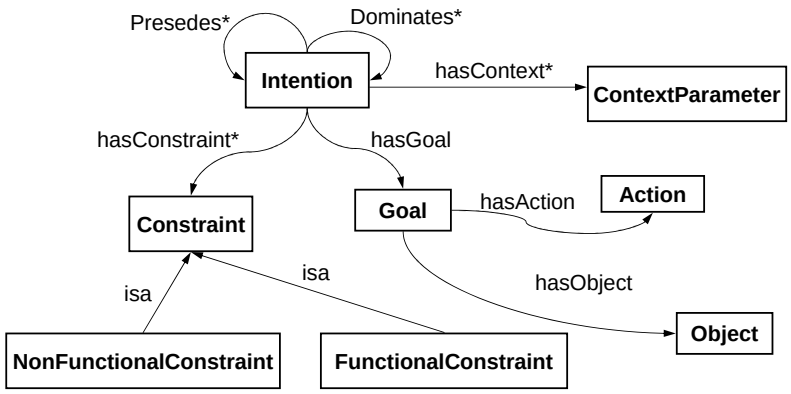

Figure 2.4 — Modèle de l'intention

2.3.4 La notion de contexte

L'approche que nous proposons dans cette thèse vise à composer automatiquement un ensemble de services et de manière que le résultat fourni à l'utilisateur soit adapté à ses besoins et à son contexte. En effet, les informations contextuelles jouent un rôle important dans la personnalisation de la composition des services pour fournir un résultat satisfaisant à l'utilisateur.

Un intérêt croissant pour la notion de contexte est partagé dans plusieurs domaines de l'informatique. Les premières utilisations du contexte dans les sciences informatiques ont été identifiées dans le domaine de l'intelligence artificielle avec les systèmes logiques du premier ordre [Weyhrauch, 1980]. Dans ce domaine, la notion de contexte est principalement présente dans deux axes de recherche : la représentation de connaissances et la logique.

Dans le domaine de l'informatique pervasive, la notion de contexte prend beaucoup d'ampleur. Schilit et Theimer [Schilit et Theimer, 1994] ont proposé une définition du contexte comme étant la localisation de l'utilisateur, les identités et les états des personnes et objets qui l'entourent. Dey [Dey et Abowd, 2000] présente le contexte comme l'état émotionnel de l'utilisateur, son centre d'attention, sa localisation, son orientation, la date et le temps où il évolue, les objets et les gens qui existent dans son environnement. Le contexte, selon Schilit, inclut la localisation et l'identité des personnes et des objets à proximité ainsi que les modifications pouvant intervenir sur ces objets [Schilit et Theimer, 1994], [Schilit *et al.*, 1994]. Il définit donc le contexte comme les changements de l'environnement physique, de

l'utilisateur et des ressources de calcul. Ward voit le contexte comme les états des environnements possibles de l'application [Ward et Jones, 1997]. En 1998, Pascoe définit le contexte comme un sous-ensemble d'états physiques et conceptuels ayant un intérêt pour une entité particulière [Pascoe, 1998]. Dey, dans [Dey *et al.*, 1999], essaie de préciser la nature des entités relatives au contexte dans la définition suivante : « *Le contexte couvre toutes les informations pouvant être utilisées pour caractériser la situation d'une entité. Une entité est une personne, un lieu, ou un objet qui peut être pertinent pour l'interaction entre l'utilisateur et l'application, y compris l'utilisateur et l'application.* » Cette définition résume toutes les autres définitions car elle est assez générique. Dey explique cette généricité du fait que les paramètres du contexte peuvent être implicites ou explicites. En fait, les paramètres du contexte peuvent être fournis par l'utilisateur ou par des capteurs comme ils peuvent parvenir d'une interprétation de ces paramètres. Winograd [Winograd, 2001] apporte plus de précision par rapport à la définition de Dey, il propose : le contexte est un ensemble d'informations. Cet ensemble est structuré et partagé ; il évolue et sert l'interprétation. Il détaille cette définition en disant : « *La considération d'une information comme contexte est due à la manière dont elle est utilisée et non à ses propriétés inhérentes* ». Il illustre cette définition par l'exemple : « *le voltage des lignes d'électricité fait partie du contexte si le système en dépend ; sinon, il ne peut être qu'un paramètre quelconque de l'environnement* ».

La notion de sensibilité au contexte concerne l'utilisation du contexte dans les applications. Un système est sensible au contexte s'il utilise le contexte pour fournir des informations et des services pertinents pour l'utilisateur, où la pertinence dépend de la tâche demandée par l'utilisateur.

Dans les applications sensibles au contexte, Dey [Dey *et al.*, 1999] propose une séparation entre la gestion du contexte et l'application, ce qui permet de développer une plateforme générique de développement et déploiement d'applications sensibles au contexte.

Pour sauvegarder une information, nous avons besoin de définir un modèle pour la décrire. Ainsi, un modèle de contexte est nécessaire pour pouvoir l'utiliser dans l'application. Les ontologies représentent une solution pour modéliser le contexte [Chen *et al.*, 2003a]. [Costa *et al.*, 2003] justifie l'utilisation des ontologies par trois arguments :

- Une ontologie permet de partager les connaissances dans un système distribué
- Une ontologie renferme des sémantiques déclaratives permettant d'élaborer des raisonnements sur les informations contextuelles.
- Avec une représentation explicite d'une ontologie commune, l'interopérabilité des applications et des équipements est assurée.

Dans notre travail, nous considérons l'existence d'une ontologie qui définit les concepts du contexte et le peuple au fur et à mesure. Nous différencions le contexte à travers deux aspects : le contexte influant le processus de composition, et le contexte influant l'exécution du service composite. Dans ce travail, nous nous concentrons sur le premier aspect. Ceci nous permet d'adapter la construction du processus de composition. L'adaptation consiste à choisir les services abstraits convenables au contexte pour générer le schéma de composition adéquat.

2.4 Conclusion

Dans ce chapitre, nous avons donné les briques de base de notre approche. Nous avons utilisé les ontologies pour la modélisation des services abstraits et des intentions. Un service abstrait est un élément de base défini au moment de la conception qui facilite la composition de services. Une intention est une structuration des besoins des utilisateurs utilisable dans un processus automatique. Le recours à la sémantique permet l'utilisation du matching sémantique et des règles de raisonnement permettant une automatisation du processus de composition. Nous avons présenté en bref le processus de composition qui est détaillé dans le chapitre suivant.

3 Approche basée sémantique pour la composition des services

3.1 Introduction

Les modèles définis pour spécifier et représenter les intentions des utilisateurs d'une part et les services abstraits d'une autre part jouent un rôle très important dans l'élaboration du service composite adéquat. Dans le chapitre précédent, nous avons introduit les modèles formels sur lesquels nous nous basons dans le travail présenté dans cette thèse. Dans ce chapitre, le processus de composition automatique est présenté en identifiant ses différents aspects. Les différentes étapes impliquées dans ce processus sont détaillées. L'étape préliminaire consiste à l'enrichissement des intentions pour exprimer les liens implicites qui lient les besoins de l'utilisateur. Ensuite, la génération du processus de composition initial est détaillée en mettant l'accent sur les aspects matching intention/service abstrait et affinité sémantique pour la connexion des services abstraits. Enfin, le schéma de composition final est généré en raffinant le schéma de composition initial. Le résultat est un plan d'exécution composé de services concrets issus de la mise en correspondance des services abstraits atomiques du schéma de composition final et les services concrets candidats.

3.2 Le processus de composition automatique

Pour composer les services, traditionnellement, il existe deux approches distinctes de conception : top-down et bottom-up [Plebani et Pernici, 2009]. La méthode Bottom-up consiste dans un premier temps à identifier l'ensemble des partenaires potentiels composé de services web (qui sont des services concrets exécutables). Ensuite, il s'agit de les connecter pour former un workflow spécifique. Quant à la méthode de conception top-down, elle commence par spécifier le processus métier (workflow) qui représente des activités abstraites non-exécutables, et, par la suite, le service concret le plus adéquat est choisi pour

chaque activité.

Nous pouvons considérer que notre approche de composition est hybride : top-down car elle commence par générer le schéma de composition identifiant les activités abstraites et le flux de contrôle entre eux sans pour autant sélectionner les services concrets exécutables ; bottom-up car le processus de composition commence par identifier l'ensemble des services abstraits contribuant à la composition. En effet, la génération du schéma de composition consiste à sélectionner les services abstraits appropriés et les combiner pour avoir le schéma de composition. Chaque service abstrait constituant le schéma de composition final est décrit par un processus atomique faisant référence à l'ensemble des services concrets potentiels qui formeraient le plan d'exécution.

Le processus de composition que nous avons défini dans notre travail inclut quatre étapes importantes consécutives : l'enrichissement du graphe des intentions, la construction du schéma de composition initial, la construction du schéma de composition final et la génération du plan d'exécution comme le montre la figure 2.1. Le graphe des intentions initial correspond à une structuration des besoins qui doivent être satisfaites. Le processus de composition prend ce type de graphe en entrée. L'activité de l'enrichissement permet de rendre explicite les relations implicites entre les intentions. Notons que dans ce travail, nous ne nous intéressons pas à la capture des intentions et à la construction du graphe. Également la collecte et le traitement des informations de contexte n'est pas notre souci dans ce travail. Le schéma de composition est un modèle de processus de services de haut niveau. Il définit le flux d'activités et leurs flux de contrôle sans identifier les services concrets à invoquer. La génération du schéma de composition initial consiste à composer un ensemble de services abstraits afin de construire un processus spécifique. Le processus de composition commence par la sélection des services abstraits (de haut niveau) correspondants aux intentions déjà identifiées. Il s'agit ensuite de les structurer dans un workflow en se basant sur la spécification des intentions. La génération du schéma de composition final est le résultat d'un processus de raffinement du schéma de composition initial jusqu'à obtenir un workflow constitué de services abstraits atomiques, que nous appelons schéma de composition final.

Le plan d'exécution est un schéma de composition exécutable, dans lequel les services concrets sont affectés aux services abstraits atomiques en tenant compte des contraintes non fonctionnelles définies dans le graphe des intentions. Quand il s'exécute, le plan d'exécution permet de rendre un service répondant aux besoins définis par le graphe des intentions initial.

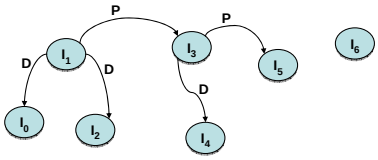

Figure 3.1 — G_0 : Graphe des intentions initial

Dans ce qui suit, nous détaillons les différentes étapes impliquées dans le processus de composition.

3.2.1 Enrichissement du graphe des intentions

L'étape de l'enrichissement du graphe des intentions a pour but d'expliciter les liens de précédence implicites entre les noeuds feuilles du graphe. Initialement, le processus de composition reçoit comme entrée le graphe des intentions. Le processus commence par l'enrichissement de ce graphe par l'application de deux règles. Nous supposons que le graphe des intentions initial est complet et comporte toutes les informations nécessaires pour l'enrichissement. D'une manière générale, l'enrichissement du graphe permet de propager le lien de précédence entre deux intentions A et B aux intentions dominées par A et B. Cela permet d'identifier les liens de précédence entre les intentions qui ne dominent pas d'autres intentions. Nous avons défini les règles d'enrichissement suivantes :

Tableau 3.1 — R1 : première règle d'enrichissement du graphe des intentions

$$Intention(?x) \wedge Intention(?y) \wedge Intention(?z) \wedge precedes(?x, ?y) \wedge dominates(?x, ?z) \rightarrow precedes(?z, ?y)$$

La règle R1 (voir tableau 3.1) permet d'identifier une intention x qui précède une intention y et une intention z dominée par x, puis ajoute un lien de précèdence entre z et y.

Tableau 3.2 — R2 : deuxième règle d'enrichissement du graphe des intentions

$$Intention(?x) \wedge Intention(?y) \wedge Intention(?z) \wedge precedes(?x, ?y) \wedge dominates(?y, ?z) \rightarrow precedes(?x, ?z)$$

La règle R2 (voir tableau 3.2) permet d'identifier une intention x qui précède une intention y et une intention z dominée par y, puis ajoute un lien de précèdence entre x et z.

En fait, le graphe des intentions est défini avec des noeuds labellisés par des noms de l'intention et des arcs orientés labellisés par le type de relation (P pour précédence et D pour

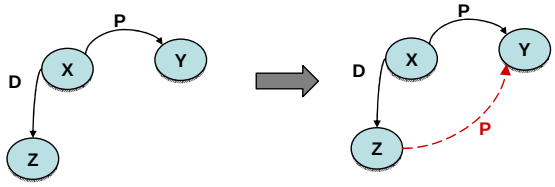

Figure 3.2 — Modification du graphe des intentions par R1

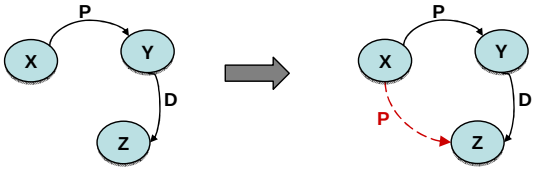

Figure 3.3 — Modification du graphe des intentions par R2

dominance). Dans la figure 3.1, nous représentons un graphe d'intentions G_0 contenant sept noeuds. Le noeud I_0 correspond à l'intention *chercher adresse de l'hôtel* Le noeud I_1 correspond à l'intention *aller à l'hôtel*, le noeud I_2 correspond à l'intention *chercher un moyen de transport*, le noeud I_3 correspond à *visiter musées*, le noeud I_4 correspond à l'intention *planifier un parcours culinaire*, le noeud I_5 correspond à l'intention *planifier de faire du shopping*, et le noeud I_6 correspond à l'intention planifier un diner d'affaire. I_1 domine I_0 et I_2 et précède I_3, I_3 domine I_4 et précède I_5.

Les règles R1 (voir figure 3.2) et R2 (voir figure 3.3) permettent d'enrichir le graphe G_0 par des liens de précédence :

— liens (I_2-I_3), (I_0-I_3) et (I_1-I_4) en identifiant le lien (I_1-I_3)
— lien (I_4-I_5) en identifiant le lien (I_3-I_5)
— lien (I_2-I_4) en identifiant le nouveau lien de précédence (I_2-I_3)
— lien (I_0-I_4) en identifiant le nouveau lien de précédence (I_0-I_3)

Nous obtenons enfin un graphe enrichi G_1 (voir figure 3.4) explicitant les relations implicites de G_0.

3.2.2 La génération du schéma de composition initial

Le schéma de composition est le workflow ou le modèle de processus métier décrivant le flux de contrôle et de données entre les activités qui le constituent. Plus précisément, il définit les fonctionnalités abstraites de services composants et leurs interactions. Le schéma de composition initial, dans ce travail, est un schéma de composition qui est composé de

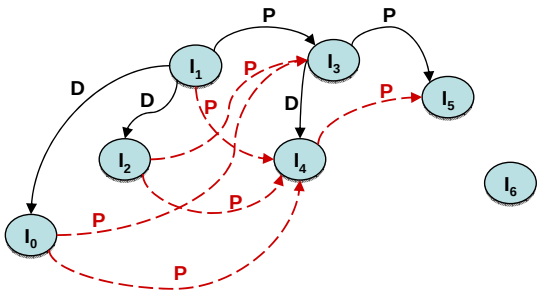

Figure 3.4 — G_1 : graphe des intentions enrichi

services abstraits et qui constitue un point de départ pour des raffinements futurs pour obtenir le schéma de composition final. La génération du schéma de composition initial a comme entrée, un graphe d'intentions enrichi (G_1)(cf. fig 3.4). Ce graphe est parcouru et les intentions sont mises en correspondance avec les services abstraits. Par ailleurs, les relations entre les intentions définies dans le graphe permettent de déduire le flux de contrôle à travers les services abstraits composant le schéma de composition. L'étude de l'affinité sémantique entre les paramètres de sortie et d'entrée des services abstraits permet de déduire le meilleur schéma de composition initial.

3.2.2.1 La construction du flux de contrôle

Le flux de contrôle consiste à avoir l'ordre des activités qui doivent être réalisées par les services. Pour identifier le flux de contrôle, nous définissons les quatre règles qui agissent sur le graphe des intentions comme suit :

- (R0) Les relations de précédence qui peuvent être déduites par transitivité sont supprimées.
- (R1) Les intentions qui ne dominent pas d'autres intentions sont mises en correspondance avec des activités qui doivent être réalisées par les services abstraits appropriés.
- (R2) La relation de précédence se traduit en une relation de séquence entre les activités : l'exécution de l'activité correspondante à l'intention précédée exige l'exécution de l'activité qui la précède.
- (R3) L'indépendance entre les intentions se traduit en une relation de parallélisme entre les activités.

Considérons le graphe des intentions enrichi G_1 représenté sur la figure 3.4. Il représente des intentions liées par des relations de dépendance et d'autres avec des connexions non spécifiées. Les intentions $\{I_0, I_1, I_2, I_3, I_4, I_5\}$ et $\{I_6\}$ sont à priori indépendantes. La règle R0 est appliquée et produit un nouveau graphe des intentions G_2 où la relation de précédence entre I_2 et I_5 (respectivement entre I_0 et I_5) est supprimée car elle peut être déduite par transitivité de la relation de précédence entre I_2 et I_4 (respectivement entre I_0 et I_4) et la relation de précédence entre I_4 et I_5 (respectivement entre I_4 et I_5).

Dans G_2, I_1 et I_3 dominent d'autres intentions, donc elles ne seront pas mises en correspondance avec des services abstraits (selon la règle R1). Dans ce cas, la relation de précédence entre I_2 et I_4 permet de déduire une relation de séquence entre s_2 et s_4 ($s_k, k \in \mathbb{N}$ étant un service abstrait). La relation de précédence entre I_0 et I_4 permet de déduire une relation de séquence entre s_0 et s_4. Les intentions I_2 et I_0 sont indépendantes donc les services s_2 et s_0 sont mises en parallèle. La relation de précédence entre I_4 et I_5 permet de déduire une relation de séquence entre s_4 et s_5 (selon la règle R2). I_6, étant indépendante, elle sera mise en correspondance avec le service s_6 qui sera mise en parallèle avec l'ensemble $\{s_0, s_2, s_4, s_6\}$ suite à l'application de la règle R3. Nous obtenons le flux de contrôle de la figure 3.5.

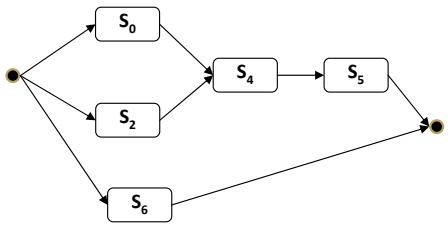

Figure 3.5 — Le flux de contrôle du schéma de composition initial

3.2.2.2 La sélection des services abstraits

La phase de sélection des services abstraits qui vont constituer le schéma de composition initial est composée des deux étapes figurées dans la figure 3.6. La première étape consiste à effectuer le matching sémantique entre les intentions et les services abstraits pour en sélectionner ceux présentent les scores de matching les plus élevés. La deuxième étape consiste à faire une deuxième sélection basée sur le degré d'affinité entre les services.

Pour le choix des services abstraits qui correspondent aux activités, nous utilisons une approche sémantique dans le but de trouver les services las plus appropriés. Notre algo-

rithme de matching compare sémantiquement l'intention au service abstrait. Le nom de l'intention est utilisé pour chercher les services abstraits susceptibles de répondre aux besoins.

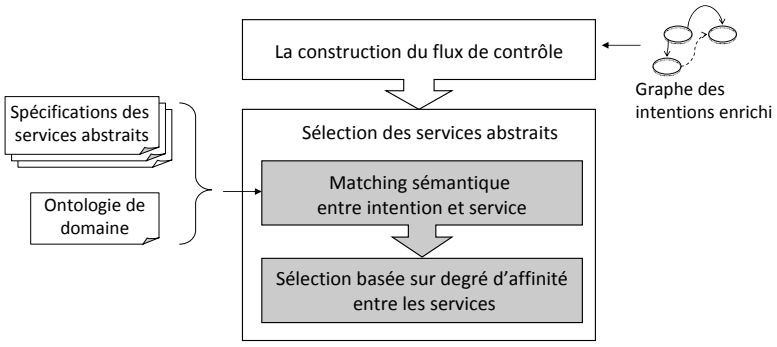

Figure 3.6 — Étapes pour la génération du schéma de composition initial

Etape1 : matching sémantique entre intention et service abstrait

Pour trouver l'ensemble des services candidats qui répondent à une intention, nous calculons le degré de matching sémantique entre le but d'une intention et le nom du service (voir figure 3.7). Ce matching est basé sur l'utilisation des ontologies et le degré de similarité sémantique. Dans notre travail, nous calculons un score de matching entre l'intention et le service qui correspond à la formule suivante :

$$score_{matching} = score_{matching\_action} + score_{matching\_object}$$

Comme nous l'avons mentionné plus haut, le but d'une intention est composé d'une action et un objet. D'une autre part, le nom d'un service décrit sa fonctionnalité en tant qu'une action et l'objet de cette action. Ainsi, pour déterminer le degré de matching entre une intention et un service ($score_{matching}$), nous calculons : (1) le score de matching ($score_{matching\_action}$) entre l'action de l'intention ($action_I$) et celle du service ($action_S$), et (2) le score de matching ($score_{matching\_object}$) entre l'objet de l'intention ($object_I$) et celui du service ($object_S$). Pour ce

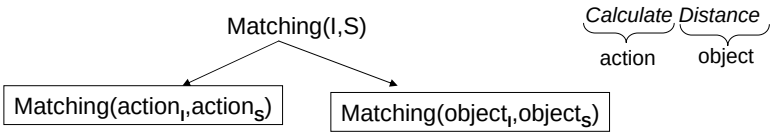

Figure 3.7 — Matching intention/service abstrait

faire, nous utilisons une ontologie de verbes et une ontologie de domaine qui représente les

objets possibles dans un domaine spécifique. Le degré de similarité représente une distance calculée en se basant sur le lien sémantique entre deux concepts dans une ontologie.

Matching action$_I$-action$_S$:

La relation de matching entre les actions est basée sur l'ontologie des verbes qui contient l'ensemble de verbes spécifiques du domaine, leurs sens et leurs relations. Un tel groupement de verbes est offert par WordNet par exemple. WordNet est un réseau sémantique de bases de données lexicales disponible gratuitement et développé en langue anglaise. Une caractéristique importante de WordNet est sa capacité de répertorier, classifier et mettre en relation de diverses manières le contenu sémantique et lexical de la langue anglaise en se basant sur les sens des informations. Chaque noeud de la base lexicale de WordNet présente un concept du monde réel et son lexique est séparé en quatre grandes catégories : les noms, les verbes, les adjectifs et les adverbes. WordNet est construit sous la forme d'une hiérarchie de concepts appelés synsets (ensemble de synonymes désignant un même sens ou un usage particulier) qui sont liés entre eux par différents types de relations sémantiques (hypéronymie, hyponymie, antonymie).

Les niveaux de matching entre les actions sont basés sur les relations suivantes :

- Exact : l'action requise est équivalente à l'action fournie.
- Synonym : l'action fournie et l'action requise ont le même sens.
- Hyponym : le sens de l'action fournie est plus général que le sens de l'action requise.
- Hypernym : le sens de l'action fournie est plus spécifique que le sens de l'action requise.

Nous pouvons donner l'exemple suivant : *Reserve* est le synonyme de *book*.

Le score de matching entre action$_I$ et action$_S$, associé à chaque niveau de matching, est défini comme [Sim et Wong, 2001] : Exact : 1, Synonym : 0.9, Hyponym : 0.7, Hypernym : 0.6.

Matching object$_I$-object$_S$:

L'évaluation du degré de similarité entre object$_I$ et object$_S$ est basée sur la comparaison entre les concepts correspondants définis dans une ontologie de domaine spécifique. Cette ontologie contient des relations hiérarchiques entre concepts (subsumption hierarchy).

Paolucci et al. dans [Paolucci *et al.*, 2002] a défini quatre niveaux de matching entre deux concepts d'une ontologie :

- Exact : le concept requis est équivalent au concept fourni
- Plug-in : le concept fourni subsume le concept requis

- Subsume : le concept requis subsume le concept fourni
- Fail : il n'y a pas de relation de subsumption entre les deux concepts

Le score de matching entre $object_I$ et $object_S$ est déterminé en se basant sur la fonction $d(C_i, C_j)$ qui mesure la distance sémantique entre deux concepts, C_i et C_j, selon l'ontologie classifiée à laquelle appartiennent les concepts. La similarité sémantique des concepts d'une ontologie de service est décrite comme suit [RADA *et al.*, 1989] :

$$S(C_i, C_j) = \frac{1}{d+1}$$

Puisque dans ce travail nous visons automatiser la composition de services pour réaliser les besoins de l'utilisateur, nous optons pour la sélection uniquement des services dont la fonctionnalité est équivalente ou plus générique que l'intention de l'utilisateur, dans le but d'éviter le cas ou le service ne rend pas la fonctionnalité requise. De ce fait, notre algorithme de matching est basé sur deux types de matching sémantique : exact et plug-in. Dans ce cas, si $C_i \equiv C_j$ (exact matching), alors $d = 0$; si $C_i \subseteq C_j$ (plug-in matching, C_i étant le concept requis et C_j le concept fourni) alors $d = 1$.

Les services présentant une valeur de similarité sémantique supérieure à un seuil spécifié par le concepteur sont ajoutés à l'ensemble S des services candidats.

Etape2 : Sélection suivant l'affinité entre les services abstraits

Une fois le flux de contrôle est déterminé et l'ensemble des services abstraits susceptibles de répondre à chaque intention sont déterminés, nous construisons les différentes alternatives possibles pour le schéma de composition initial. Par exemple, pour l'intention I_2, deux services abstraits s_2 et s'_2 peuvent être sélectionnés. Pour l'intention I_4, deux services abstraits s_4 et s'_4 peuvent être sélectionnés. Chacune des intentions I_5 et I_6 lui correspond un seul service abstrait s_5 et s_6. Pour ce cas de figure, nous obtenons quatre alternatives pour le schéma de composition initial comme le montre la figure 3.8.

Le choix entre les différentes alternatives sera basé sur la qualité de la connexion sémantique entre les services (la relation entre les paramètres d'entrée et sortie). Ces paramètres sont des concepts d'une ontologie. Donc il s'agit de calculer le degré de similarité sémantique entre l'ensemble des paramètres de sortie $s_x.Out$ du service s_x et l'ensemble des paramètres d'entrée $s_y.In$ du service s_y, s_x et s_y étant deux services composés en séquence. Ainsi, s_x et s_y sont liés sémantiquement en se basant sur une fonction de matching $Sim(out, in)$, avec $out \in s_x.Out$ et $in \in s_y.In$. Dans [Lécué et Léger, 2006], les quatre types de fonctions du matching sémantique proposés par [Paolucci *et al.*, 2002] sont considérés pour vérifier

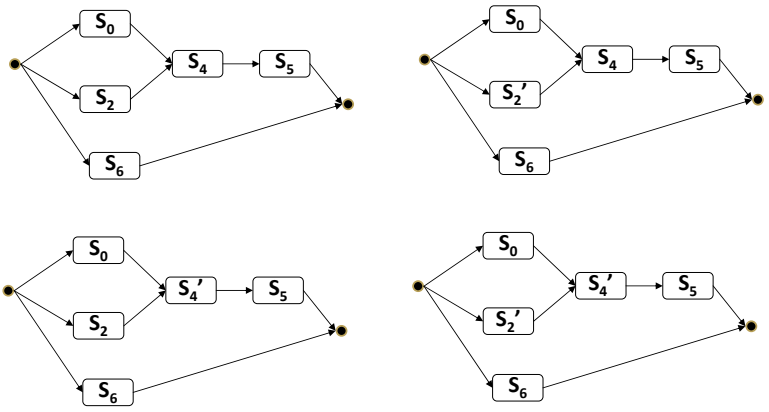

Figure 3.8 — Différentes alternatives pour le schéma de composition initial

Tableau 3.3 — Les fonctions de matching sémantique décrites par *Sim*

Match type	*Exact*	*Plug-in*	*Subsume*	*Fail*
$Sim(out, in)$	1	$\frac{2}{3}$	$\frac{1}{3}$	0
Logic meaning	$out \equiv in$	$out \subset in$	$out \supset in$	Otherwise

la similarité sémantique entre le concept out et le concept in. La similarité sémantique est évaluée par la fonction *Sim* (voir tableau 3.3) qui donne le degré de lien entre les paramètres des services. Par exemple, la relation de matching *Plug-in* veut dire qu'un paramètre de sortie d'un service s_x est un sous-concept d'un paramètre d'entrée du service s_y tandis que la relation de matching *Subsume* veut dire qu'un paramètre de sortie du service s_x est un super-concept d'un paramètre d'entrée du service s_y.

Dans notre approche, le calcul de la similarité sémantique à cette étape n'est pas utilisé pour déduire quels services dont les sorties alimentent les entrées d'un autre service ; en d'autres termes, quels sont les services qui dépendent des autres services. En effet, la relation de séquence est déjà établie, c'est qu'il y a au moins une paire de l'ensemble des paramètres des entrées de s_y et des sorties de s_x qui ont une relation de dépendance sémantique (*Exact*, *plug-in* ou *Subsume*). En d'autres termes :

$$\exists out_j \in s_x.Out, \exists in_i \in s_y.In \text{ tel que } Sim(out_j, in_i) > 0$$

Notre objectif est de trouver le meilleur schéma de composition selon un critère d'optimisation. Ce critère est la qualité de connexion sémantique entre les services. Donc nous considérons le degré de matching entre les paramètres d'entrée et de sortie des services pour dégager ceux qui offrent les valeurs maximales de similarité sémantique.

Considérons que $s_x.Out = \{out_1, out_2, ..., out_n\}$ est l'ensemble des sorties de s_x, et $s_y.In =$

$\{in_1, in_2, ..., in_m\}$ est l'ensemble des entrées de s_y.

Pour calculer la similarité sémantique entre s_x et s_y, il faut établir le lien entre $s_x.Out$ et $s_y.In$. Pour cela, la valeur de similarité de chaque paramètre d'entrée in_i de s_y avec chaque paramètre de sortie out_j de s_x est mesurée pour en retenir enfin la valeur maximale. En effet, la paire (out_j, in_i) qui présente la valeur de similarité maximale constitue le couple pour lequel la valeur de out_j serait consommée par in_i. Ensuite, la somme de ces valeurs maximales est calculée et est divisée par le nombre des inputs de s_y.

L'affinité sémantique qui donne le degré de dépendance entre s_x et s_y est donnée par la fonction *semAff* :

$$semAff(s_x.Out, s_y.In) = \frac{1}{m} \sum_{i=1}^{m} \max_{j=1}^{n} Sim(out_j, in_i) \tag{3.1}$$

avec $(0 \leq semAff(s_x.Out, s_y.In) \leq 1)$

Rappelons que la fonction *semAff* (3.1) est utilisée pour calculer le degré de l'affinité sémantique entre uniquement deux services séquentiels. En pratique, il est possible qu'un service s_y soit précédé par plus qu'un service ; il est nécessaire donc de prendre en compte l'ensemble des outputs des services qui précédent s_y. Considérons l'ensemble des services S parallèles qui précédent s_y. Il faut trouver pour chaque paramètre d'entrée de s_y un paramètre de sortie de l'ensemble des paramètres de sortie des services contenus dans S, avec lequel la similarité est maximale. La figure 3.9 montre un exemple de correspondance entre les paramètres de sortie des services de S et les paramètres d'entrée de s_y. L'entrée in_3 de s_y, par exemple, réalise la similarité maximale avec la sortie out_2 de S.

En effet, on ne sait pas à priori quel service de S qui fournit le paramètre de sortie qui alimente un paramètre d'entrée de s_y. L'algorithme 3.1 permet de trouver les paires *(paramètre de sortie, paramètre d'entrée)* qui correspondent à la valeur de similarité maximale, et permet en conséquence de déduire, pour chaque service de S, l'ensemble des paramètres de sortie qui seront consommés par les paramètres d'entrée de s_y. Ceci permet enfin de calculer l'affinité sémantique entre chaque service de S et le service s_y.

Considérons $s_y.In$ l'ensemble des paramètres d'entrée de s_y, et $S.Out$ l'ensemble des paramètres de sortie des services de l'ensemble S. L'étape de matching sémantique entre intention et services abstraits peut donner deux services candidats s_k et $s_{k'}$, qui offrent les mêmes possibilités de matching pour un ensemble de paramètres d'entrée de s_y : $\{Inputs_k\}$. Donc l'affinité sémantique entre $s_{k'}$ et s_y est calculée ainsi :

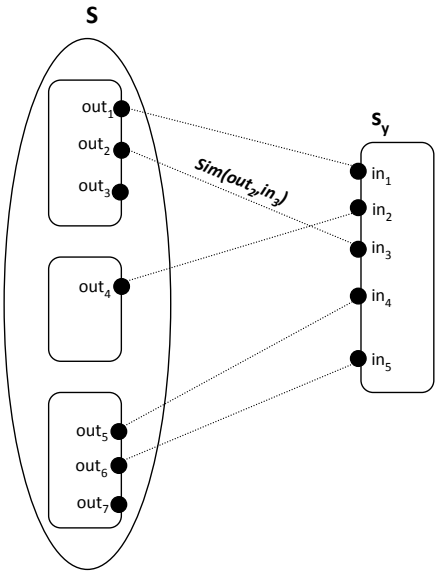

Figure 3.9 — Correspondance entre les paramètres de sortie des services de S et les paramètres d'entrée de s_y

Algorithme 3.1 — Algorithme de calcul de l'affinité sémantique entre deux services composés en séquence

Input : s_y,S
Output :
foreach *input* $in_i \in s_y.In$ **do**
  $Sim_{max} \leftarrow 0$
  foreach *output* $out_j \in S.Out$ **do**
    if $Sim(out_j, in_i) > Sim_{max}$ **then**
      $Sim_{max} \leftarrow Sim(out_j, in_i)$
      $out_{max} \leftarrow out_j$
    end
  end
  foreach *service* $s_k \in S$ **do**
    if $out_{max} \in s_k.Out$ **then**
      $somme_{maxSim[s_k]} \leftarrow somme_{maxSim[s_k]} + Sim_{max}$
      $nbInput_k \leftarrow nbInput_k + 1$
      //le nombre des entrées de s_y à correspondre avec les sorties de s_k
      $\{Inputs_k\} \leftarrow \{Inputs_k\} \cup in_i$
    end
  end
end
foreach *service* $s_k \in S$ **do**
  $semAff(s_k, s_y) \leftarrow somme_{maxSim[s_k]} / nbInput_k$
end

$$semAff(s_{k\prime}, s_y) = \frac{1}{nbInput_k} \sum_{i=1}^{nbInput_k} \max_{j=1}^{p} Sim(out_j, in_i)$$

avec $\{out_1, ..., out_p\}$ est l'ensemble des outputs de $s_{k\prime}$ à matcher avec l'ensemble $\{Inputs_k\}$ des inputs de s_y.

Après avoir calculé l'affinité sémantique de toutes les paires des services séquentiels, l'étape suivante consiste à dégager les meilleures correspondances pour déduire le meilleur schéma de composition initial. Le meilleur schéma de composition est obtenu en maximisant la somme des valeurs des affinités sémantiques des paires des services en séquence comme suit :

$$maximize \sum semAff(s_i, s_j)$$

avec s_i et s_j sont deux services liés par une relation de séquence.

La figure 3.10 présente un exemple qui montre les différentes alternatives possibles pour obtenir le schéma de composition.

Considérant cet exemple, la valeur de l'affinité sémantique entre s_{01} et s_{41} (cas (1)) présente la valeur la plus grande par rapport aux autres alternatives ((2), (3) et (4)). Cependant l'alternative (1) est écartée, et c'est l'alternative (2) qui est retenue car la somme des valeurs des affinités sémantiques $semAff(s_{01}, s_{42}) + semAff(s_{42}, s_5) + semAff(s_2, s_{42}) = 0.8 + 0.7 + 0.6 = 2.1$ est la valeur maximale. Ainsi, les services s_{01} et s_{42} sont les services candidats retenus.

Cette sélection basée sur le degré d'affinité sémantique entre les services permet par conséquent d'établir le flux de données final entre les services sélectionnés, c'est à dire faire le lien entre les entrées/sorties compatibles.

3.2.3 La génération du schéma de composition final

Dans cette étape, nous utilisons le schéma de composition initial (voir figure 3.11) issu de l'étape précédente. Il sera encore raffiné, et d'autres services abstraits de granularité plus fine seront sélectionnés itérativement jusqu'à obtenir un schéma de composition composé de services abstraits atomiques. Au cours de ce processus, la partie *process* (voir figure 2.3) de chaque service abstrait faisant partie du schéma de composition initial est analysée (voir figure 3.11). Deux cas peuvent se présenter :

i) Si la partie *process* du service abstrait est atomique, il n'y aura plus de raffinement de

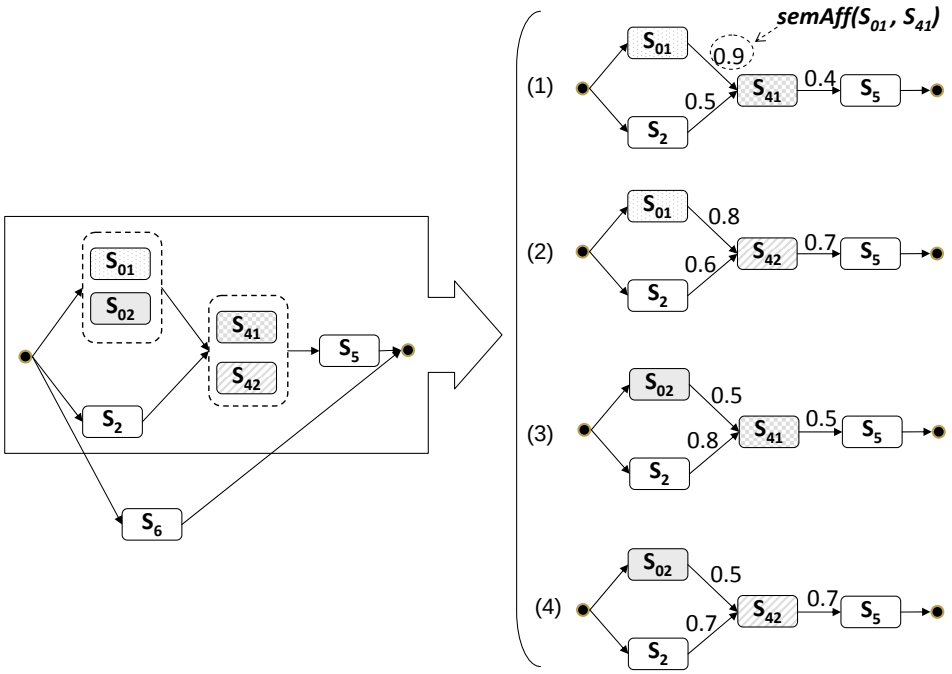

Figure 3.10 — Représentation graphique des différentes alternatives possibles du schéma de composition initial

ce service abstrait. La collection des services concrets OWL-S candidats, associée à ce service abstrait sera analysée durant la phase de génération du plan d'exécution.

ii) Si la partie *process* du service abstrait est composite, chacun de ses sous-processus (qui sont tous des processus simples) est analysé : avec ces processus simples, une recherche est effectuée pour obtenir les services abstraits correspondants.

Le traitement d'un processus simple est traité comme présenté dans la figure 3.11 : Le processus simple est substitué par la partie *process* du service abstrait correspondant. Le service correspondant est celui qui matche le processus simple selon un degré de similarité. Si la partie *process* de ce service est un processus atomique, on est placé dans le cas i). S'il est composite, ce processus simple est étendu à un processus composite. Dans ce cas, la procédure décrite en ii) est appliquée.

Le raffinement du schéma de composition initial est basé sur la substitution des processus simples (qui font partie des services abstraits) par les services abstraits correspondants. Pour que cette substitution soit possible, il faut que le service abstrait substituant ait une fonctionnalité équivalente à la fonctionnalité fournie par le processus simple.

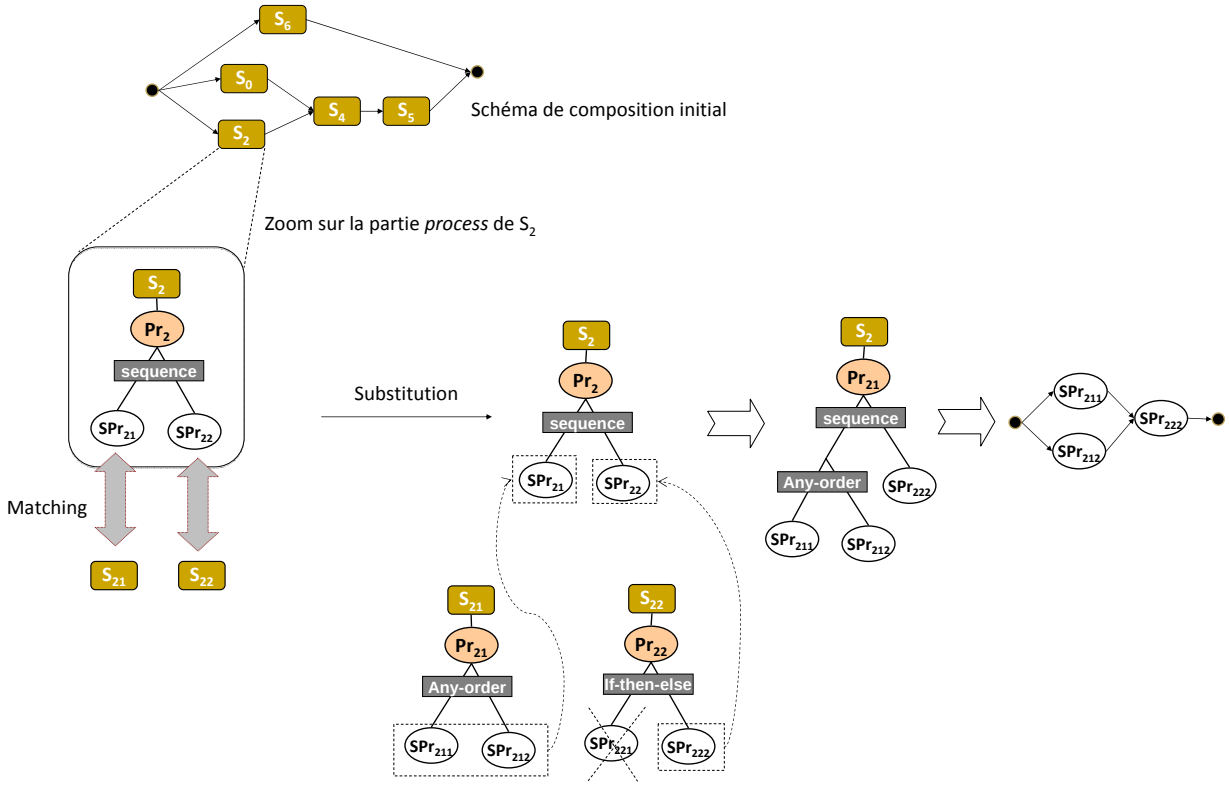

Figure 3.11 — Raffinement du schéma de composition initial : substitution des sous-processus par les services abstraits correspondants

Le flux de contrôle du schéma de composition est mis à jour par le processus de raffinement. En effet, la composition des processus composites qui alimentent itérativement le schéma de composition est spécifiée par la structure de contrôle (*control construct*) définie dans le service abstrait (voir figure 2.3). Certaines de ces structures de contrôle indiquent la présence d'un ordre spécifique des activités d'un processus tel que *Sequence* et *Any-Order*. D'autres structures de contrôle appellent au choix d'une seule activité parmi un ensemble d'activités telles que *Choice* et *If-then-else*. En effet, certains services abstraits proposent des alternatives pour satisfaire la fonctionnalité du service abstrait. Chacune de ces alternatives est influencée par une contrainte fonctionnelle spécifique ou des données contextuelles. Si le choix de l'alternative ne peut pas se faire pendant la composition car elle est en relation avec les données de l'exécution , dans notre approche, cette condition est conservée dans le schéma de composition pendant l'étape de raffinement des services abstraits. Durant le processus de composition, le sous-processus approprié est choisi. Les informations fournies par la spécification des intentions et du contexte sont utilisées pour diriger la sélection des activités alternatives.

3.2.3.1 Matching sémantique

Nous nous appuyons sur le concept de similarité que nous avons utilisé dans la section 3.2.2.2. Nous évaluons le degré de similarité entre le processus simple et le service abstrait correspondant. Pour déterminer le degré de similarité entre le sous-processus spr et le service abstrait s, nous calculons la distance sémantique entre :

i) le nom du sous-processus et le nom du service abstrait.

ii) les paramètres d'entrée et de sortie du sous-processus et du service abstrait.

La condition (C) pour que le service s soit considéré comme matching possible pour le sous-processus spr est l'existence de paramètres d'entrée et de sortie de s dont la distance sémantique avec les paramètres d'entrée et de sortie de spr est non nulle d'une part. D'une autre part, la distance sémantique entre les noms du service s et du sous-processus spr doit être non nulle. Cette condition peut s'exprimer comme suit :

$$C \equiv \begin{cases} \forall in_k^{spr} \in spr.In, \exists in_p^s \in s.In \text{ tel que } d(in_k^s, in_p^{spr}) > 0 \\ \forall out_k^{spr} \in spr.Out, \exists out_p^s \in s.Out \text{ tel que } d(out_k^s, out_p^{spr}) > 0 \\ d(s.Object, spr.Object) > 0 \\ d(s.Action, spr.Action) > 0 \end{cases}$$

avec $spr.In$ et $s.In$ les ensembles des paramètres d'entrée, $spr.Out$ et $s.Out$ les ensembles des paramètres de sortie, $s.Action$ et $s.Object$ constituent le nom du service abstrait et $spr.Action$ et $spr.Object$ constituent le nom du sous-processus.

Soit $\mathbb{S}$ l'ensemble des services s qui vérifient la condition C pour un sous-processus spr. Nous parcourons l'ensemble $\mathbb{S}$ en calculant la distance sémantique entre s et spr suivant la formule suivante :

$D(spr, s) = \sum_{j=1}^{|spr.In|} d(in_j^{spr}, in_j^s) + \sum_{j=1}^{|spr.Out|} d(out_j^{spr}, out_j^s) + d(s.Object, spr.Object) + d(s.Action, spr.Action)$

L'ensemble $\mathbb{S}$ est mis à jour, il contient désormais le service ayant la distance minimale avec le sous-processus spr.

3.2.3.2 Utilisation du contexte

Les opérations que nous venons de décrire sont exécutées pour que tous les sous-processus des services abstraits constituant le schéma de composition initial soient traités. Ceci produit un nouveau schéma de composition qui doit être raffiné de la même manière. Ces opérations sont répétées jusqu'à l'obtention d'un schéma de composition constitué de

processus des services abstraits atomiques, appelé schéma de composition final. Le choix des services est conditionné par les éléments du contexte présents dans la description de l'intention d'origine. Les services conditionnels permettent d'évaluer une condition pour pouvoir choisir l'alternative (sous-processus) qui répond le mieux à cette condition. La condition est généralement associée à un élément de contexte, par exemple l'existence de capteurs de présence dans un bâtiment. Par ailleurs, les contraintes fonctionnelles définies dans le graphe des intentions sont prises en compte lors du raffinement. Par exemple, l'unité de mesure d'un équipement qui donne la température.

Algorithme 3.2 — Algorithme de génération du schéma de composition final

```
   Input : CS : Initial composition schema (an OWL-S file)
   Output : Final composition schema
 1 //variables : s, s1, spr, pr, pr1
 2 foreach s1 in CS do
 3     if process pr1 of s1 is composite then
 4         foreach sub-process spr of pr1 do
 5             Search for and load service s that corresponds to spr;
 6             if process pr of s is composite then
 7                 get the control construct of pr;
 8                 if control construct==if-then-else or control construct==choice then
 9                     use intention.FC and contextualData to select appropriate
                       sub-process;
10                     substitute spr by the selected sub-process;
11                 end
12                 else
13                     substitute spr by pr of s;
14                 end
15             end
16             else
17                 if process pr of s is atomic then
18                     substitute spr by s;
19                 end
20             end
21         end
22     end
23 end
```

Le processus de génération du schéma de composition final est présenté dans l'algorithme 3.2.

Pour illustrer le fonctionnement de l'algorithme, nous traitons par exemple le cas du service abstrait s_2 du schéma de composition initial (voir figure 3.11). L'algorithme 3.2 vérifie que la partie processus pr_2 du service s_2 se compose de deux sous-processus en séquence (ligne 3). Pour chaque sous-processus (ligne 4), l'algorithme cherche le service correspon-

dant (ligne 5). Une fois le service s_{21} et le sous-processus spr_{21} sont mis en correspondance à travers le calcul de la distance sémantique et le degré de similarité, la partie procesus du service s_{21} est examinée (lignes 6, 7 et 8). La structure de contrôle du processus de s_{21} correspond à *any-order* ce qui permet la substitution de spr_{21} par le processus du service s_{21} (ligne 13). L'algorithme s'exécute de la même manière pour les sous-processus spr_{211} et spr_{212}, et donne une correspondance avec des services abstraits atomiques s_{211} et s_{212}. Le traitement du sous-processus spr_{22} donne une correspondance avec le service abstrait s_{22}. La structure de contrôle de s_{22} correspond à *if-then-else* ce qui permet la substitution de spr_{22} par le sous-processus spr_{222} de s_{22} (lignes 9 et 10). Le choix du spr_{222} est guidé par la prise en compte du contexte. L'intention I_2 *chercher moyen de transport* fait intervenir l'élément de contexte *moyen de transport* qui peut être : *bus* ou *métro*. Le choix de spr_{222} est conditionné par la disponibilité du moyen de transport *métro*. L'algorithme traite de la même manière le sous-processus spr_{222} et donne une correspondance avec un service atomique s_{222}. Le traitement du service s_2 est terminé puisque son raffinement donne un fragment du schéma de composition final constitué de processus issus de services atomiques comme le montre la figure 3.11.

Le schéma de composition final obtenu est présenté par la figure 3.12.

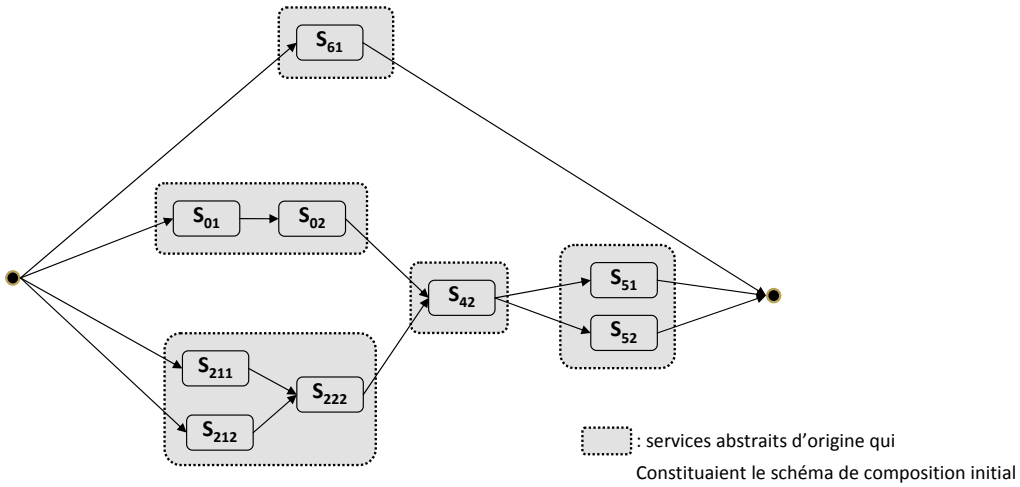

Figure 3.12 — Le schéma de composition final

3.2.4 Génération du plan d'exécution

Une fois le schéma de composition final est généré, l'étape suivante dans le processus de composition consiste à déterminer les services web qui vont constituer le plan d'exécution. Cette étape utilise les contraintes non-fonctionnelles fournies par la spécification des intentions pour sélectionner les services appropriés. Rappelons qu'un service abstrait atomique

est associé à un ensemble de services concrets OWL-S. L'implémentation de chaque service concret OWL-S et les détails pour accéder à et invoquer le service réel sont spécifiés par la partie *Grounding*. Cet ensemble de services concrets OWL-S constitue une collection de services web ayant une fonctionnalité commune et présentant des propriétés non fonctionnelles différentes (par exemple : fournisseurs différents, valeurs de QoS différentes, etc). Ces propriétés non fonctionnelles sont spécifiées dans la partie profil de chaque service concret. L'ensemble des services concrets candidats associés à chaque service abstrait atomique peut être fixé lors de la conception du service abstrait, ou il peut être déterminé à travers un mécanisme d'enregistrement permettant ainsi aux fournisseurs de services de rejoindre ou quitter l'ensemble à tout moment.

Il est nécessaire de sélectionner le service concret qui permet d'implémenter le service abstrait en tenant compte des contraintes non-fonctionnelles définies dans le graphe des intentions, comme le montre la figure 3.13. Rappelons que le service abstrait qui est mis en correspondance à une intention I pendant la phase de la génération du schéma de composition initial pourrait être raffiné pendant la phase de génération du schéma de composition final. Ceci donnerait une combinaison d'un ensemble de services $\mathbb{S}_{\mathbb{I}}$ représentant un fragment de processus qui réalise l'intention I. De ce fait, il ne s'agit pas de considérer localement le meilleur service concret (de point de vue QoS offerte) qui peut réaliser un service abstrait de $\mathbb{S}_{\mathbb{I}}$. En d'autres termes, l'ensemble des services concrets appropriés qui seront sélectionnés pour chaque service abstrait de $\mathbb{S}_{\mathbb{I}}$ doivent satisfaire la (ou les) contrainte non fonctionnelle exprimée dans I pour avoir le meilleur fragment de processus associé à I. Il est donc nécessaire de propager la contrainte non fonctionnelle de l'intention I à l'ensemble des services concrets qui la réalisent car le processus de raffinement peut donner un fragment de processus qui réalise une seule intention.

Considérons l'exemple d'une contrainte qui concerne le temps d'exécution : nous pouvons calculer le temps d'exécution total du fragment de processus associé à une intention donnée par l'agrégation des valeurs des temps d'exécution des services qui le composent. Plus généralement, la fonction d'agrégation dépend de l'attribut de la QoS et du flux de contrôle de la composition. Nous avons considéré dans le tableau 3.4 les fonctions d'agrégation pour les attributs : temps d'exécution, prix et disponibilité. q_T, q_P, q_D désignent respectivement les valeurs des attributs de QoS : Temps d'exécution, Prix et Disponibilité. L'ensemble des s_i représente les services concrets qui correspondent aux n services abstraits qui constituent la composition (dont la structure est soit séquentielle soit parallèle).

Prenons le cas de l'intention I_2 (voir graphe G_0 dans la figure 3.1) qui contient la

Tableau 3.4 — Fonctions d'agrégation pour quelques attributs de QoS

Attribut QoS	Structure de contrôle de la composition	
	Séquence	Parallèle
Temps d'exécution	$\sum_{i=1}^{n} q_T(s_i)$	$\max_{i=1}^{n}\{q_T(s_i)\}$
Prix	$\sum_{i=1}^{n} q_P(s_i)$	$\sum_{i=1}^{n} q_P(s_i)$
Disponibilité	$\prod_{i=1}^{n} q_D(s_i)$	$\prod_{i=1}^{n} q_D(s_i)$

contrainte non-fonctionnelle : *temps d'exécution < x secondes*. L'intention I_2 est réalisée par un fragment de processus contenant trois services (voir figure 3.13) : s_{211}, s_{212} et s_{222}. Le temps d'exécution de ce fragment ne doit pas dépasser x secondes donc il faut que :

max(temps d'exécution de s_{211}, temps d'exécution de s_{212})+ temps d'exécution de s_{222} < x secondes.

Plus généralement, pour le temps d'exécution, la formule utilisée correspond au maximum des sommes des temps d'exécution des branches en parallèle. Dans la figure 3.13 nous expliquons comment nous tenons compte de la contrainte non fonctionnelle de l'intention I_2 pour sélectionner les services concrets correspondants aux services abstraits s_{211}, s_{212} et s_{222}.

Le résultat de cette étape donne un plan d'exécution composé de services web exécutables qui réalisent les besoins de l'utilisateur.

En tant que tel, une attention particulière doit être prise pour contrôler comment un service concret est choisi lorsque plusieurs services concrets sont associés à un service abstrait atomique.

3.3 Conclusion

Dans ce chapitre, nous avons détaillé le processus de composition. Afin de construire une composition, nous avons commencé tout d'abord par l'enrichissement du graphe des intentions. Ensuite, nous avons présenté la démarche pour la construction des flux de contrôle et de données du schéma de composition initial. Nous avons utilisé les techniques de matching sémantique en nous appuyant sur le concept de similarité sémantique. Enfin la génération du schéma de composition final est présentée à travers un algorithme qui tient compte des paramètres du contexte. Nous avons par la suite expliqué comment le plan d'exécution est généré en tenant toujours compte des intentions fournies. En effet, cette étape utilise les contraintes non-fonctionnelles fournies par la spécification des intentions pour sélection-

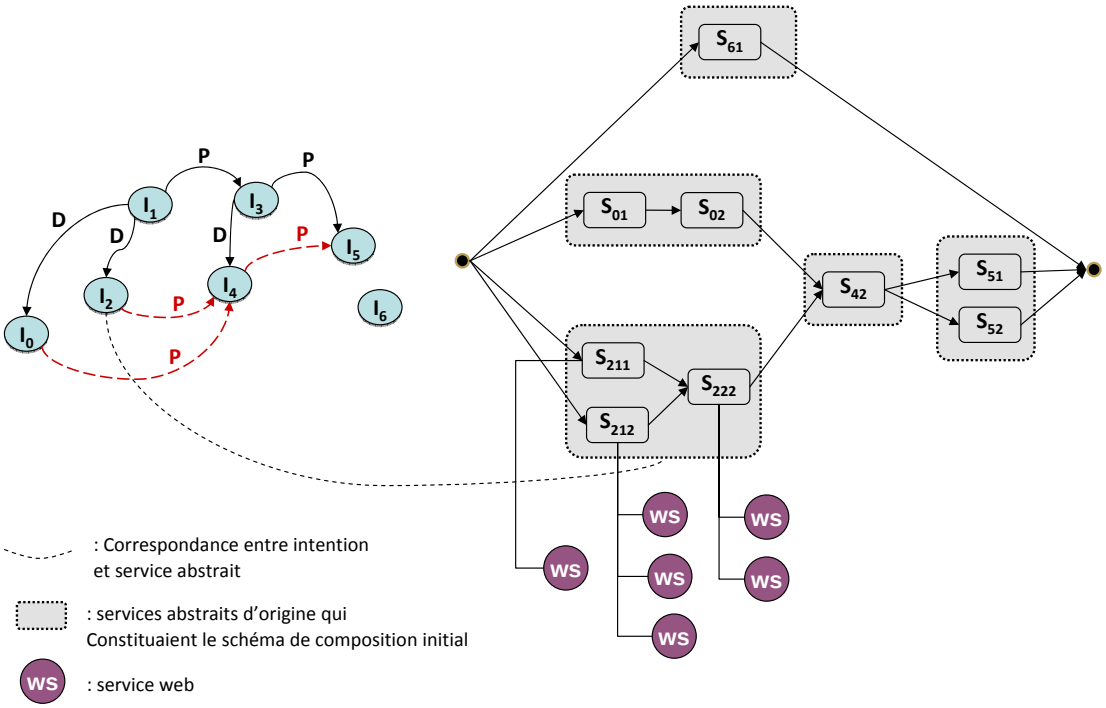

Figure 3.13 — Propagation des contraintes non fonctionnelles

ner les services appropriés car il est nécessaire de sélectionner le service concret qui permet d'implémenter le service abstrait en tenant compte des contraintes non-fonctionnelles définies dans le graphe des intentions. La mise en œuvre du processus de composition fait l'objet du chapitre suivant.

4 Implantation et évaluation de notre approche de composition des services

4.1 Introduction

Dans le but de tester et de valider notre travail, nous avons implanté le processus de composition décrit dans les chapitres précédents. Dans ce chapitre, nous présentons l'architecture de l'environnement de composition en détaillant les différents éléments le constituant. Ensuite nous présentons les outils que nous avons développés pour la composition des services. Nous commençons par présenter les choix techniques et logiciels que nous avons faits pour l'implantation de nos contributions théoriques. Puis, un cas d'étude qui illustre l'approche proposée et l'utilisation de l'environnement de composition est détaillé. Enfin une évaluation de l'environnement est proposée.

4.2 Mise en œuvre de l'environnement de composition

4.2.1 Architecture de l'environnement de composition

Dans ce qui suit, nous présentons l'environnement de composition suivant l'approche définie dans la figure 2.1. Nous détaillons un ensemble de modules organisés dans une architecture.

Notre implantation de l'environnement de composition, illustrée sur la figure 4.1 est constituée des éléments suivants :

— *Le module de traitement des intentions* : le point d'entrée de ce module est un fichier OWL décrivant la spécification des intentions d'un utilisateur. Ce module applique les règles d'enrichissement du graphe des intentions décrites dans le chapitre 2. Il permet parsuite d'extraire toutes les paires des intentions feuilles (qui ne dominent

pas d'autres intentions) qui sont liées par une relation de précédence. Cette extraction est effectuée grâce à l'application de la règle SQWRL (voir table 4.1) :

Tableau 4.1 — Extraction des intentions feuilles

$$Intention(?x) \wedge Intention(?y) \wedge precedes(?x,?y) \wedge Intention(?z) \wedge Intention(?t) \wedge Intention(?xd) \wedge Intention(?yd) \wedge dominates(?xd,?z) \wedge dominates(?yd,?t) \wedge sqwrl:makeSet(?s1,?x) \wedge sqwrl:groupBy(?s1,?y) \wedge sqwrl:makeSet(?s2,?y) \wedge sqwrl:groupBy(?s2,?x) \wedge sqwrl:makeSet(?s1d,?xd) \wedge sqwrl:makeSet(?s2d,?yd) \wedge sqwrl:difference(?s3,?s1,?s1d) \wedge sqwrl:difference(?s4,?s2,?s2d) \wedge sqwrl:element(?ex,?s3) \wedge sqwrl:element(?ey,?s4) \rightarrow sqwrl:select(?ex,?ey)$$

— *Le module de construction du flux de contrôle* : ce module construit le schéma de composition sans tenir compte des services abstraits qui le constitueraient. En d'autres termes il génère sa structure de contrôle générale. Il prend comme entrée toutes les intentions non dominantes et le lien qui existe entre chaque deux intentions. Il se base sur les règles de composition que nous avions définies pour générer le flux de contrôle du schéma de composition initial.

— *Le module de matching sémantique* : ce module effectue le matching sémantique entre deux concepts d'une ontologie selon les niveaux de matching (exact, subsume, plug-in, fail). Il utilise l'ontologie de domaine traité. Il utilise le raisonneur Pellet [1] pour inférer les relations entre les concepts. Ce module est utilisé pour le matching entre les intentions et les services abstraits et pour le calcul de l'affinité sémantique entre les services abstraits. Ce module utilise une ontologie de domaine.

— *Le module de sélection des services abstraits pour le schéma de composition initial* : Ce module est appelé par le module de construction du flux de contrôle. Il assure la sélection des services abstraits qui constitueraient le schéma de composition initial. Pour ce fait : il implémente les deux étapes définies dans la section 3.2.2.2 du chapitre 2, à savoir : la mise en correspondance de chaque intention avec l'ensemble des services abstraits adéquats et la sélection basée sur le calcul du degré d'affinité entre les services abstraits en utilisant le dépôt de services abstraits. Pour effectuer le matching sémantique entre une intention et un service abstrait d'une part, et entre les paramètres d'entrée et de sortie d'une autre part (pour calculer le degré d'affinité entre les services), ce module de sélection s'appuie sur le module de matching sémantique. Le module de sélection des services abstraits établit à la fin le flux des données entre les services sélectionnés.

— *Le module de raffinement du schéma de composition initial* : Le point d'entrée de ce module est le schéma de composition initial. Il effectue la phase de raffinement en sélection-

1. Pellet : http ://www.mindswap.org/2003/pellet/

Tableau 4.2 — Outils, langages et technologies utilisés dans l'environnement de composition

Tâche	Outils/langages/technologies
Programmation	Java 1.6
Gestion d'ontologies	OWL 1, SWRL, Protégé 3.2.1, Pellet 2.0, SQWRL, OWL-S

nant itérativement d'autres services abstraits de granularité plus fine. Il fait appel au module de matching sémantique pour effectuer le matching. Il génére le schéma de composition final. Par ailleurs, ce module met à jour le flux des données puisqu'il substitue la partie process des services abstraits du schéma de composition par d'autres services.

— *Le module de sélection des services concrets* : ce module génère le plan d'exécution en parcourant le fichier OWL représentant le schéma de composition final. Il exploite la base des services web qui est en relation avec le dépôt des services abstraits.

4.2.2 Outils, langages et technologies utilisés

Les modules de l'architecture de composition ont été implémentés en Java. L'implantation de l'environnement de composition a été particulièrement complexe du fait que nous avons eu recours à divers outils et bibliothèques externes, qui sont également écrits en Java. Nous avons aussi utilisé plusieurs langages de modélisation et les technologies qui leur sont associées.

Le Tableau 4.2 résume tous les langages, les outils, les bibliothèques et les technologies qui interviennent dans la conception des modèles utilisés et dans l'exécution de la plateforme de composition.

Dans ce qui suit, nous présentons les outils pour le traitement des ontologies OWL et des services abstraits. Une des raisons du succès et de l'utilisation grandissante des technologies du Web Sémantique est l'existence de nombreux outils pour la gestion des ontologies OWL. En effet, il existe des bibliothèques, des interfaces de programmation (API), des éditeurs, des moteurs d'inférence et de règles, etc. qui facilitent la création et l'édition d'ontologies et de règles, leur accès depuis un programme, l'exécution du raisonnement et le traitement de règles. Une grande partie de ces logiciels sont open sources.

Édition d'ontologies L'éditeur Protégé [2] [Knublauch *et al.*, 2004], développé par l'Université de Stanford en collaboration avec l'Université de Manchester, est le standard de facto pour la création et l'édition d'ontologies OWL. Son code source, libre, est écrit en Java, et

2. Disponible sur http ://protege.stanford.edu/

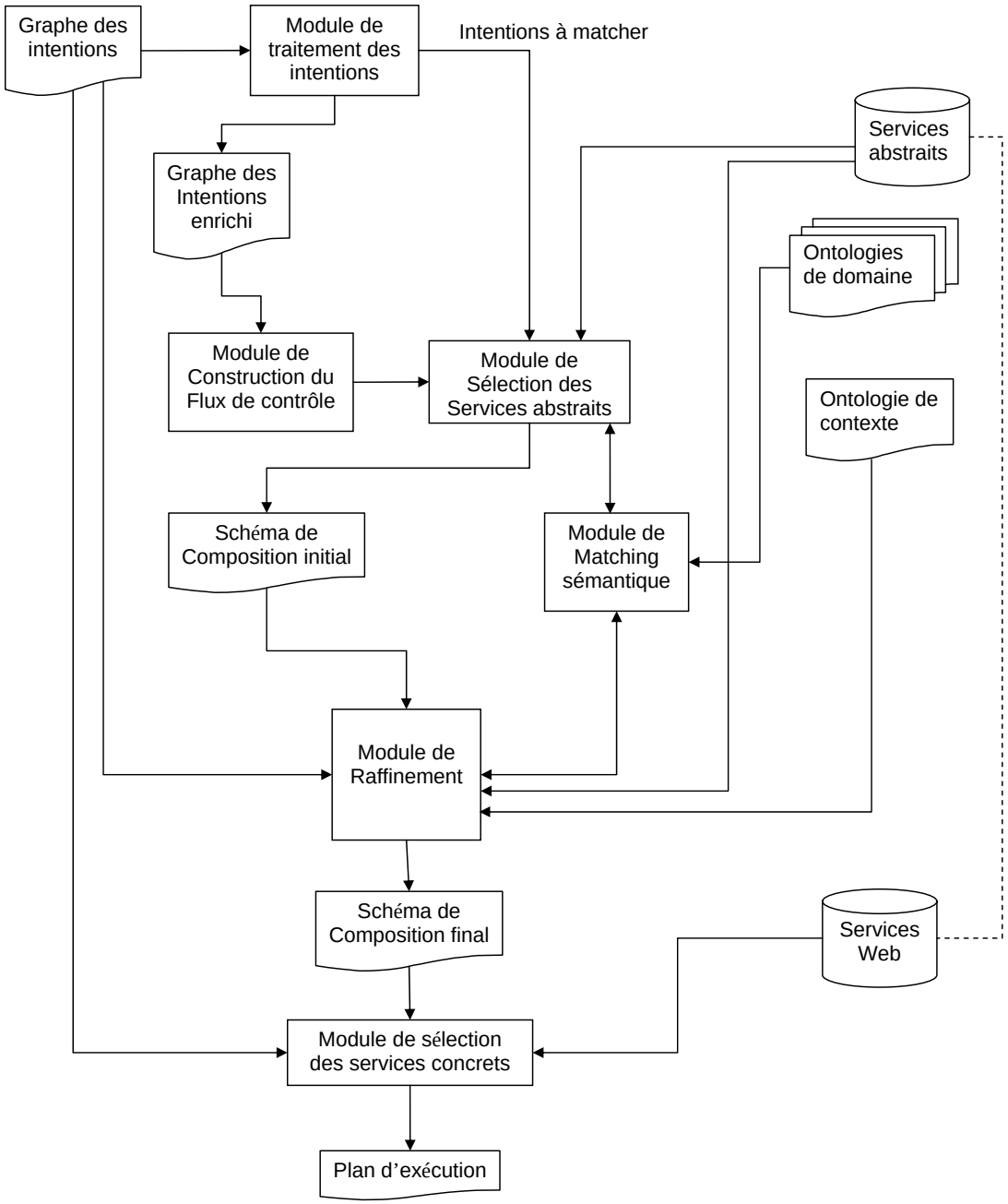

Figure 4.1 — Architecture de l'environnement de composition

il admet des extensions sous forme de *plugins*. Il existe de nombreux *plugins,* par exemple pour la visualisation des ontologies et pour l'édition des règles SWRL associées.

Pour la création et l'édition de nos services abstraits qui sont décrits avec l'ontologie OWL-S, nous avons utilisé l'outil OWL-S Editor. Cet éditeur est un outil dont l'objectif est de permettre de concevoir facilement et de manière intuitive des services OWL-S. *OWL-S*

Editor[3] est implémenté en tant qu'un *Plugin* open source pour l'éditeur Protégé.

Nous avons utilisé la version 3.2.1 de Protégé pour la manipulation des services abstraits car c'est la version la plus récente qui est compatible avec le *Plugin* de OWL-S Editor.

APIs pour le traitement des ontologies OWL et OWL-S

Ces bibliothèques logicielles permettent un accès par programme aux ontologies OWL ; elles fournissent des fonctions qui permettent de créer une ontologie, de la lire, de créer le modèle en mémoire correspondant, de le modifier, de l'enregistrer, etc. Ces bibliothèques sont implantées avec le langage Java. Les deux bibliothèques principales pour le traitement des ontologies OWL sont OWL API[4] et Protégé-OWL API.

OWL API est l'implantation de référence pour la création, la manipulation et la sérialisation d'ontologies OWL. OWL-API est un projet libre mené par l'université de Manchester qui supporte pleinement OWL 2. Il fournit également plusieurs interfaces pour l'accès à des moteurs d'inférence d'une façon transparente. Protégé-OWL API est un *plugin* de Protégé qui est utilisé pour l'accès aux ontologies OWL dans les versions 3.x de l'éditeur. Il peut être utilisé par des programmes externes comme une API Java, indépendamment de Protégé. Tout comme OWL-API, il permet l'accès à des moteurs d'inférence externes.

OWL-S API[5] fournit une API Java qui permet à un programme de créer, lire, écrire, et exécuter des services décrits en OWL-S. La bibliothèque fournit un moteur d'exécution basique *ProcessExecutionEngine* qui peut exécuter les processus atomiques utilisant WSDL et les processus composites qui utilisent les structures de contrôle tels que *Choice*, *Sequence*, *AnyOrder*, *Split*, et *Split-Join*. L'exécution des processus qui utilisent des structures de contrôle conditionnelles comme *IfThenElse*, *RepeatUntil*, ou *RepeatWhile* est aussi supportée. L'API est conçue pour être extensible de sorte que d'autres formalismes (logiques) puissent être intégrés. L'API est distribuée avec le raisonneur Pellet. D'autres raisonneurs peuvent être intégrés. La structure des classes de l'API a été conçue pour correspondre étroitement aux définitions des ontologies OWL-S, spécifiquement les versions OWL-S 1.2 et 1.1. En outre, l'API fournit un support quasi complet pour les tâches de gestion pour les standards RDF et OWL telles que l'ajout, la suppression et l'interrogation de triplets.

3. http ://owlseditor.semwebcentral.org/ ou https ://bintray.com/kemlg/owlsutils/com.sri.owlseditor
4. Disponible sur http ://owlapi.sourceforge.net/
5. http ://on.cs.unibas.ch/owls-api/

4.3 Cas d'étude

4.3.1 Description de l'environnement

L'étude de cas concerne un bâtiment intelligent qui se définit comme un bâtiment à haute efficacité énergétique, intégrant dans la gestion intelligente du bâtiment des équipements consommateurs (climatiseur, lampe), des équipements producteurs (panneaux photovoltaïques) et des équipements de stockage (batteries). Le bâtiment intelligent est équipé d'une multitude de capteurs (capteurs de température, de présence, de luminosité) destinés à évaluer l'état du bâtiment et des entités qui s'y trouvent. Le but est de le rendre le plus adapté possible aux personnes qui y travaillent. Cela va de la maîtrise de l'énergie, aux exigences de confort.

Le bâtiment comporte plusieurs sources d'énergie utilisées notamment pour la régulation de la température du bâtiment et la production de l'énergie électrique. Pour la régulation de la température, le chauffage du bâtiment se fait à l'aide d'un puits géothermique et d'un puits canadien. L'eau venant du puits géothermique est utilisée pour les radiateurs. L'air issu du puits canadien est injecté dans le bâtiment pour stabiliser sa température. Pour la production de l'énergie électrique, le bâtiment contient une surface importante de panneaux photovoltaïques. L'énergie produite par les panneaux est consommée en priorité par le bâtiment. Le surplus de production est soit stocké dans les batteries, soit réinjecté dans le réseau. Pour évaluer l'état du bâtiment, les capteurs fournissent les données nécessaires. Les capteurs de présence permettent de détecter les personnes dans les différents secteurs du bâtiment. Les détecteurs de luminosité permettent de déterminer le degré de luminosité dans les différentes pièces. Les capteurs de température extérieurs et intérieurs permettent de donner la différence entre les températures à l'intérieur et l'extérieur du bâtiment.

Suite à l'évaluation de l'état du bâtiment, des actions de reconfiguration sont nécessaires pour optimiser la consommation en énergie du bâtiment. Différents actionneurs sont utilisés. Des actionneurs sont disposés sur les lampes, sur les stores des fenêtres, sur les climatiseurs, sur les vannes de distribution d'eau, sur celles de distribution d'air, et sur les disjoncteurs électriques. Les actionneurs sur les lampes et les stores permettent de réguler la luminosité, d'augmenter ou de diminuer l'intensité des lampes ou contrôler l'ouverture des stores suivant le degré de luminosité. Par ailleurs les actionneurs sur les climatiseurs et les vannes permettent de réguler le chauffage suivant les préférences des utilisateurs et suivant les températures à l'intérieur et à l'extérieur du bâtiment.

Une infrastructure logicielle est mise en place en se basant sur la technologie des ser-

vices (par exemple services web) permet à un administrateur du bâtiment d'interagir avec des équipements (capteurs et actionneurs) qui sont accessibles à travers un réseau d'accès/communication. Les différents équipements sont capables de répondre à des requêtes émanant des logiciels utilisés pour la gestion du bâtiment.

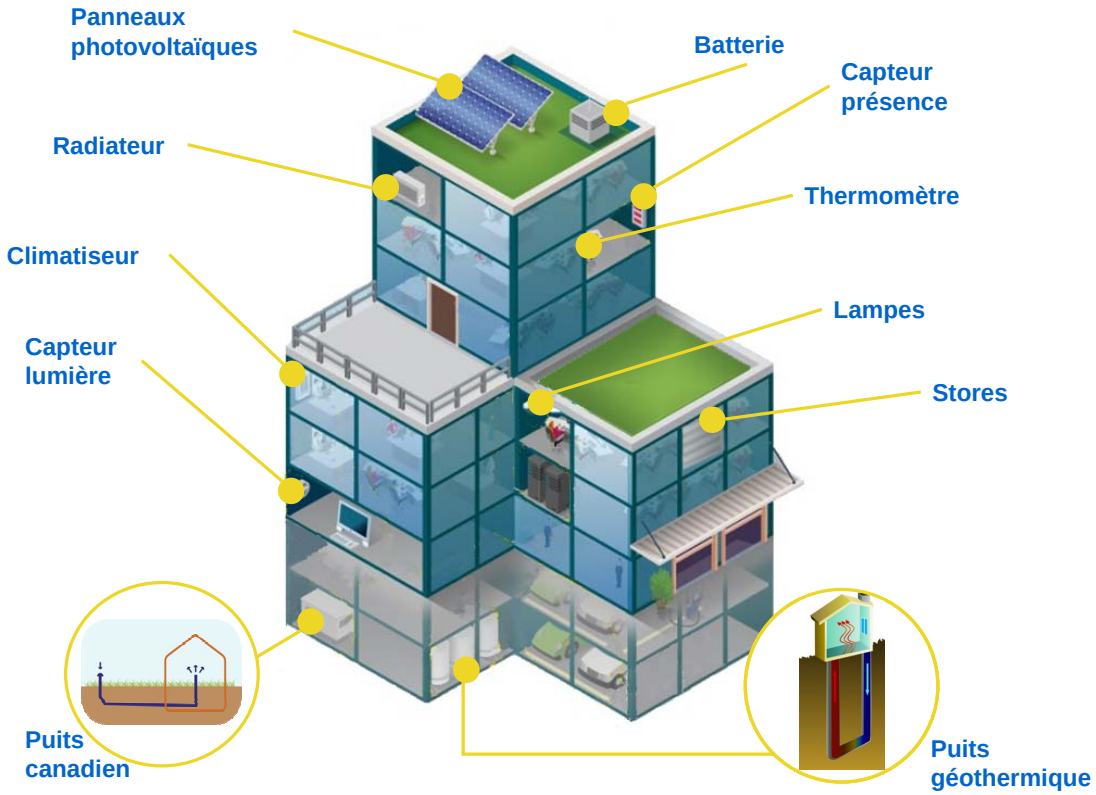

Figure 4.2 — Bâtiment intelligent

4.3.2 Graphe initial des intentions du cas d'étude

Pour appliquer notre approche, nous considérons les acteurs suivants : l'administrateur du bâtiment et les utilisateurs du bâtiment. Nous considérons particulièrement une salle du bâtiment intelligent. La salle est équipée d'un ensemble de lampes, d'un radiateur, un climatiseur et une fenêtre dont les stores sont ouvrables automatiquement.

Nous considérons que pour l'administrateur son objectif principal est : *Optimiser la consommation d'énergie*. Concernant les utilisateurs du bâtiment, nous considérons que leur besoin principal est de travailler dans des conditions les plus confortables possibles. La réalisation de l'objectif de l'administrateur passe par la réalisation de l'intention I_1 dont le but est *Optimize energy consumption* en tenant compte d'un certain nombre d'hypothèses qui garantissent le confort des utilisateurs du bâtiment surtout au niveau température et luminosité.

Comme point de départ, nous considérons le graphe des intentions G_0 de la figure 4.3. L'intention I_1 (dont le but est *Optimize energy consumption*) domine les intentions I_3 dont le but est *Regulate luminosity* et I_5 dont le but est *Regulate temperature*. Pour pouvoir réguler la luminosité, l'intention I_2 dont le but est *Calculate luminosity* précède l'intention I_3. Pour pouvoir réguler la température, l'intention I_4 dont le but est *Calculate temperature* précède l'intention I_5 qui domine les intentions I_6 dont le but est *Regulate air-conditioning* et I_7 dont le but est *Regulate heating system*. Ce graphe s'accompagne avec un certain nombre de précisions données par le tableau 4.3. L'intention I_2, par exemple, possède deux contraintes fonctionnelles : l'unité et la précision. L'unité est le lumen, et la précision de calcul est fixée à 0.5.

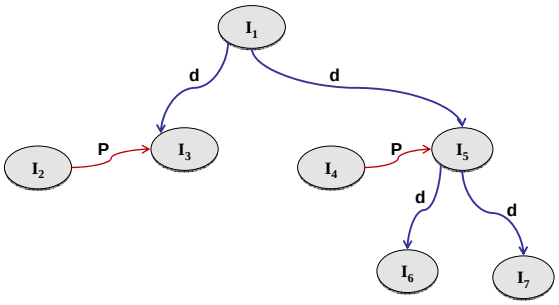

Figure 4.3 — Graphe des intentions : Optimiser la consommation d'énergie dans une pièce du bâtiment intelligent

I_2 possède par ailleurs une contrainte non fonctionnelle qui exige que le temps d'exécution du service qui réalise le but de I_2 ne dépasse pas 10 unités de temps. L'intention I_3 possède deux contraintes fonctionnelles qui fixent la borne inférieure (1200) et supérieure (1800) de la luminosité dans la pièce. Pour I_3, l'élément de contexte associé c'est *Window Blinds*. L'intention I_4 possède deux contraintes fonctionnelles : l'unité (Celsius) et la précision (0.5). L'intention I_5 possède deux contraintes fonctionnelles qui fixent la borne inférieure (15) et supérieure (25) de la température dans la pièce. Les intentions I_6 et I_7 sont en relation avec les éléments de contexte *Heater* et *Air Conditioner*.

4.4 Application de l'approche sur le cas d'étude

Dans le paragraphe suivant, nous appliquons les différentes étapes de notre approche en montrant comment l'environnement de composition est utilisé pour générer un schéma de composition.

Tableau 4.3 — Détails des intentions du graphe G_0

ID	But	Contraintes fonctionnelles	Contraintes non fonctionnelles	Paramètres de contexte
I1	Optimize energy consumption			
I2	Calculate luminosity	-Precision=0.5 -unit=lumen	Response Time<10	
I3	Regulate luminosity	-LuminosityMax=1800 -LuminosityMin=1200		Manual Window Blinds
I4	Calculate temperature	-Precision=0.5 -Unit=celsius		
I5	Regulate temperature	-TempMax=25 -TempMin=15		
I6	Regulate air-aonditioning			-Heater -Air Conditioner
I7	Regulate heating system			-Heater -Air Conditioner

4.4.1 Application des règles d'enrichissement

Le module de traitement des intentions prend en entrée le graphe G_0 et exécute les règles d'enrichissement. Les règles d'enrichissement de la section 3.2.1 sont appliquées au graphe G_0 qui permet d'expliciter les relations de précédence implicites. Nous obtenons le graphe de la figure 4.4 qui contient une relation de précédence entre I_4 et I_6 et une autre entre I_4 et I_7.

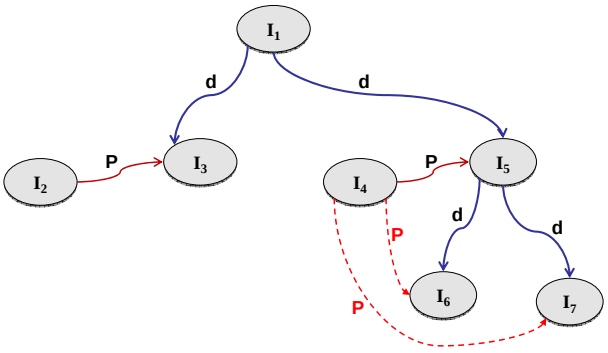

Figure 4.4 — G_1 : Graphe des intentions enrichi

4.4.2 Génération du schéma de composition initial :

Le module de construction du flux de contrôle reçoit le graphe G_1 et construit le flux donné par la figure 4.5. Le module de sélection des services abstraits fournit les méthodes nécessaires pour trouver des services abstraits qui peuvent être utilisées par l'application. Il fait appel au module de matching sémantique qui retourne une liste de services abstraits qui correspondraient aux intentions feuilles du graphe G_1. Ce matching est calculé comme détaillé dans la section 3.2.2.2 étape 1. Par exemple, pour l'intention I_4, l'action est *Calculate* et l'objet est *temperature*. L'ensemble des services abstraits découverts sont :
S_{41} : *Evaluate Weather*
S_{42} : *Calculate Temperature*
S_{43} : *Determine Temperature*
S_{44} : *Calculate HeatDegree*
Le calcul du score de matching des services S_{41}, S_{42}, S_{43}, S_{44} donne respectivement : 0.7, 2, 1.7, 1.5. Le seuil pour le score de matching étant fixé à 1.5, uniquement les services S_{42}, S_{43}, S_{44} seront admis au niveau suivant pour la sélection.

Le module de sélection des services abstraits sélectionne aussi les services abstraits adéquats pour les intentions I_6 et I_7. C'est la partie profile du service abstrait qui est analysée pour trouver les services convenables. Pour I_6, deux services abstraits S_{61} et S_{62} sont sélectionnés. Pour l'intention I_7, ce sont les services abstraits S_{71} et S_{72} qui sont sélectionnés. Les intentions I_6 et I_7 sont précédées par l'intention I_4. Nous avons l'ensemble $\{S_{42},S_{43},S_{44}\}$ qui précède l'ensemble $\{S_{61},S_{62},S_{71},S_{72}\}$. L'algorithme 3.1 est appliquée pour le calcul de l'affinité sémantique sur les différentes combinaisons possibles entre les éléments des deux ensembles. D'une autre manière, le degré d'affinité entre les paramètres de sortie des services abstraits de l'ensemble $\{S_{42},S_{43},S_{44}\}$ et les paramètres d'entrée des services abstraits de l'ensemble $\{S_{61},S_{62},S_{71},S_{72}\}$ est calculé, ce qui explore 12 combinaisons pour ce fragment du schéma de composition initial. Pour les intentions I_2 et I_3, le même traitement est effectué par le module de sélection des services abstraits. Le schéma de composition généré est présenté dans la figure 4.5. Il met en évidence les flux de données établis entre les services qui le constituent. Nous avons annoté la figure 4.5 par les noms courts (ou symboliques) des services qui constituent le schéma de composition. Par exemple, le service *calculate temperature* génère la valeur de la température qui est une entrée pour les services *Regulate HeatingSystem* et *Regulate Air-Conditioning*. De même, le service *Determine Luminosity* génère une valeur d'entrée pour le service *Regulate Luminosity*. Cette valeur représente le degré de luminosité.

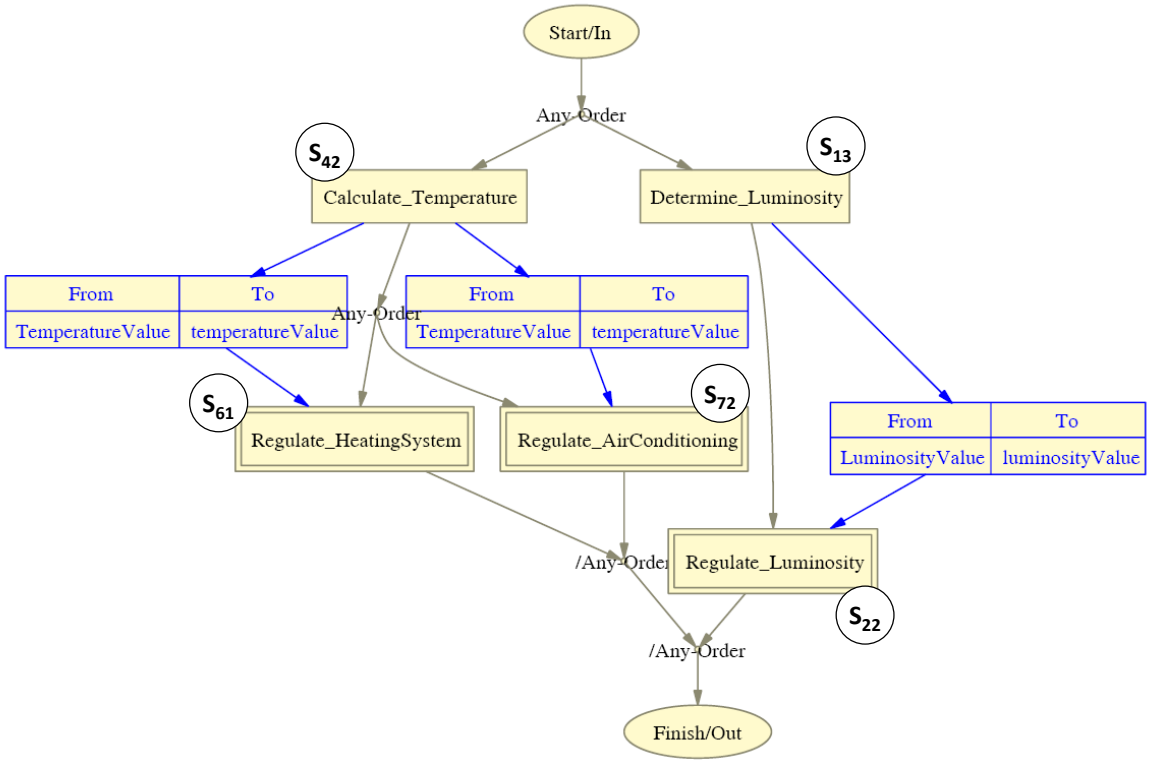

Figure 4.5 — Schéma de composition initial

4.4.3 Génération du schéma de composition final

Le module de raffinement prend en entrée le schéma de composition initial (voir figure 4.5) pour le raffiner en d'autres services abstraits de granularité plus fine. Dans notre cas, le service abstrait *Determine Luminosity* (S_{13}) est atomique. Il n'est donc pas raffiné. Le service *Regulate Luminosity* (S_{22}) est un service composite dont la partie *process* Pr_{22} est constituée de trois sous-processus : *Check Presence* (SPr_{221}), *TurnOff Lamp* (SPr_{222}) et *Adjust Luminosity* (SPr_{223}) organisés comme le montre la figure 4.6.

En fait, le fonctionnement du service S_{22} commence par la vérification de l'état de présence dans la pièce ce qui conditionne l'extinction de la lampe ou l'ajustement de la luminosité. Le module de raffinement appelle le module de matching sémantique pour chercher les services abstraits correspondants à SPr_{221}, SPr_{222}, et SPr_{223}. Plusieurs alternatives sont trouvées, et à travers le score de matching, les meilleurs services abstraits sont sélectionnés. Les services *Check Presence* (S_{221}) et *TurnOff Lamp* (S_{222}) (qui correspondent à SPr_{221} et SPr_{222} respectivement) trouvés sont atomiques. Le service *Adjust luminosity* (S_{223}) est un service composite qui sera raffiné à son tour. La partie *process* du service S_{223} est constituée de deux sous-processus : *Reduce Lamp Intensity* (SPr_{2231}) et *Increase luminosity* (SPr_{2232}) orga-

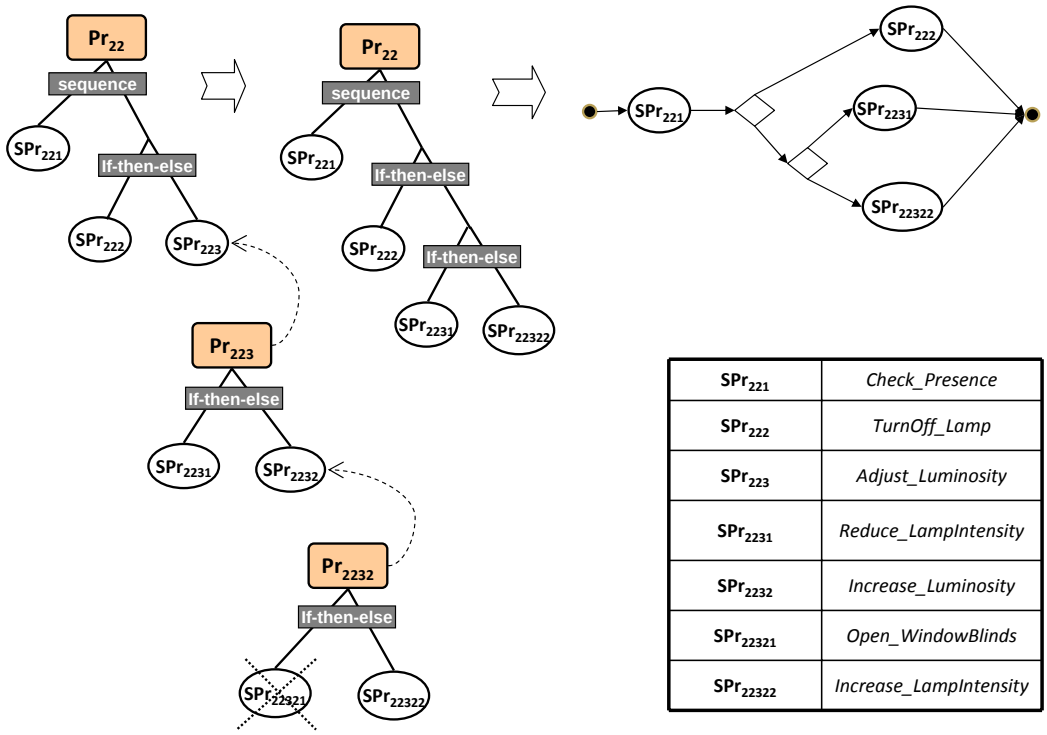

Figure 4.6 — Raffinement du service *Regulate Luminosity* (S_{22})

nisés comme le montre la figure 4.6. L'ajustement de la luminosité se fait par rapport à la contrainte exprimée par l'intention I_3 qui fixe la limite maximale de luminosité. Ce service entraîne soit une réduction de l'intensité de la lampe, soit une augmentation de la luminosité dans la pièce (si la luminosité <*seuilMin* alors augmenter luminosité, sinon si luminosité > *seuilMax* alors réduire l'intensité de la lampe). Cette condition est conservée dans le schéma de composition puisqu'elle dépend d'une donnée de contexte déterminée lors de l'exécution, à savoir la valeur de luminosité. Le module de raffinement appelle le module de matching sémantique pour traiter les sous-processus SPr_{2231} et SPr_{2232}. Les services correspondants trouvés sont S_{2231} qui est atomique et S_{2232} qui est composite. La partie *process* du service S_{2232} est constituée de deux sous-processus : *Open Window Blinds* (SPr_{22321}) et *Increase LampIntensity* (SPr_{22322}) qui représentent deux alternatives possibles. Le choix de l'une de ces deux alternatives est conditionné par l'existence de stores de fenêtres automatiques dans la pièce. Si les stores automatiques existent, l'augmentation de la luminosité pourrait se faire par l'ouverture automatique des stores, sinon elle se fait avec l'augmentation de l'intensité de la lampe. Dans le cas de ce scénario, le contexte indique que les stores des fenêtres dans la pièce ne sont pas automatiques (voir tableau 4.3). Ainsi, le sous-processus *Increase LampIntensity* (SPr_{22322}) est choisi pour substituer SPr_{2232} (voir figure 4.6). La correspondance de *Reduce Lamp Intensity* (SPr_{2231}) et *Increase LampIntensity* (SPr_{22322}) avec les

services abstraits donne deux services abstraits atomiques. Ceci termine le raffinement du service abstrait *Regulate Luminosity* (S_{22}).

Le service abstrait *Calculate Temperature* (S_{42}) est atomique tandis que les services *Regulate heating System* (S_{61}) et *Regulate Air-Conditioning* (S_{72}) sont composites. Le module de raffinement et le module de matching sémantique permettent de raffiner ces services jusqu'à obtenir des services atomiques.

En fait le fonctionnement du service S_{61} (respectivement S_{72}) commence par la vérification si la salle est une salle de manipulation, si oui l'ajustement du chauffage (respectivement la climatisation) dans la salle de manipulation dépend de l'existence de l'équipement *Heater* (respectivement *AirConditioner*) exprimé par l'intention I_6 (respectivement I_7). Le schéma de composition final est donné par la figure 4.7.

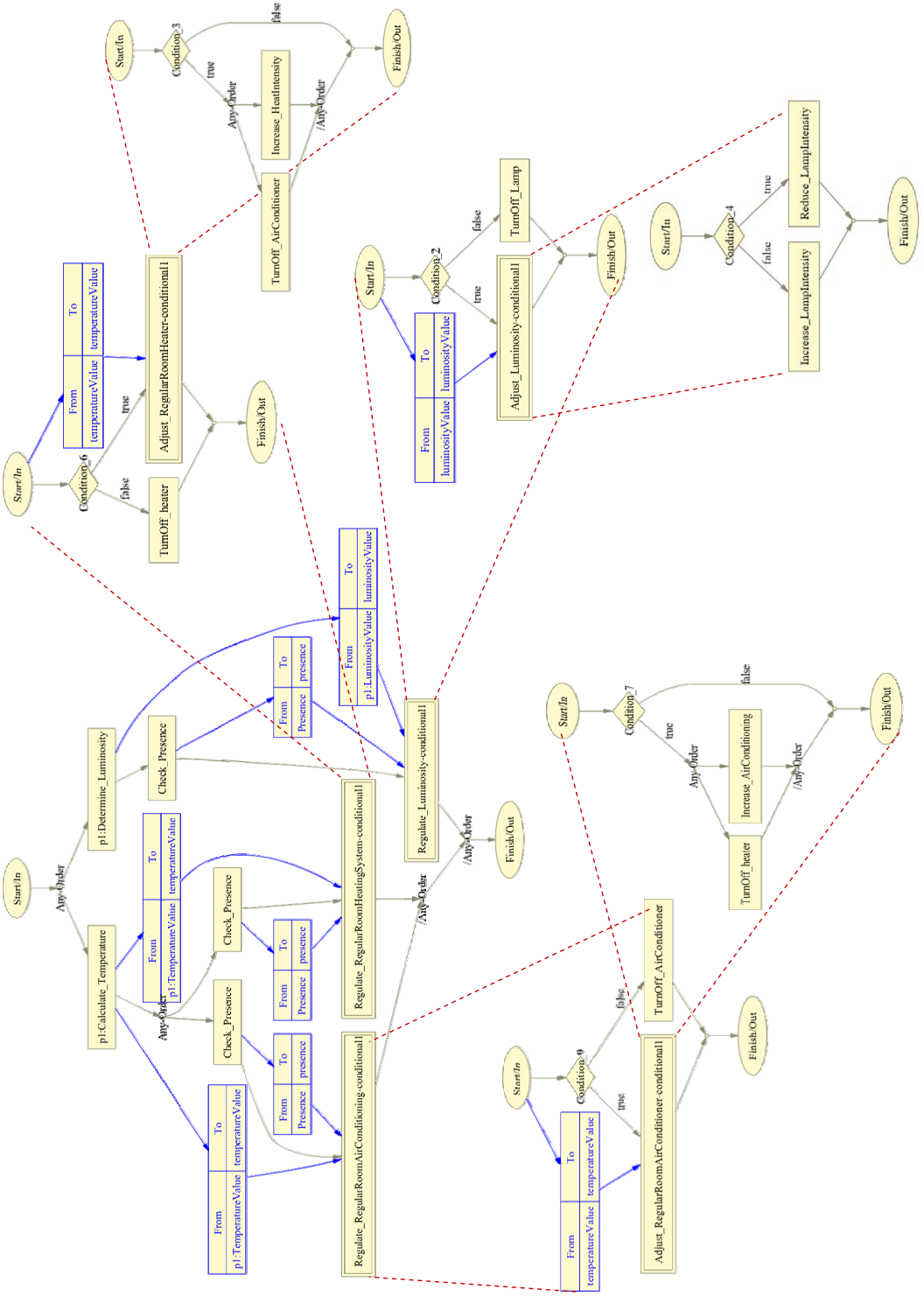

Figure 4.7 — Raffinement complet du schéma de composition initial

4.5 Évaluation

Dans cette section, nous évaluons le fonctionnement de notre environnement de composition en relation avec le cas d'étude du bâtiment intelligent.

4.5.1 Évaluation du temps d'exécution

Afin d'évaluer le temps d'exécution de notre environnement, nous avons mené des tests préliminaires. Ces expérimentations ont été faites sur une machine de 1,87 GHz CPU, et 3 Go de mémoire. nous mesurons trois temps de référence :

— le temps T1 d'enrichissement du graphe des intentions qui contient le résultat du traitement de la règle SWRL d'enrichissement.

— T2 le temps de génération du schéma de composition initial qui contient la mise en correspondance des intentions avec les services abstraits, la construction du flux de contrôle, et la construction du flux de données par matching sémantique.

— T3 le temps de génération du schéma de composition final qui contient le raffinement du schéma de composition initial.

A partir du cas d'étude Smart building nous avons généré des graphes des intentions dont la taille varie de 7 à 20 nœuds. Nous considérons qu'un graphe des intentions pour notre cas peut difficilement dépasser les 20 nœuds. Pour chaque taille de graphe, nous traçons les temps T1, T2 et T3 en secondes avec un intervalle de confiance de 10%. Nous obtenons la figure 4.8 avec les résultats globaux pour notre expérimentation (T1 en gris, T2 en noir, T3 en blanc). Nous remarquons que jusqu'à 11 nœuds, le temps d'exécution global reste sous la barre des 4 secondes. Entre 12 et 16 nœuds, le temps global est en dessous de 8 secondes (sauf pour un graphe contenant 13 nœuds). A partir de 17 nœuds, le temps dépasse 10 secondes. En fait, l'enrichissement (T1) qui revient à une recherche d'un sous graphe dans un graphe constitue une activité consommatrice en temps. Par rapport à la génération du schéma de composition final (T3), c'est la profondeur du raffinement qui fait que cette étape est consommatrice en temps. Pour la génération du schéma de composition initial, T2 dépend fortement du nombre de services abstraits disponibles correspondant aux différentes intentions exprimées. Dans notre cas, un maximum de 15 services abstraits étaient disponibles pour chaque intention.

Il est à noter que le raisonnement sémantique est coûteux en termes de temps de traitement. Malgré cela, nous estimons que ce temps est acceptable pour les utilisateurs finaux qui souhaitent obtenir de nouvelles compositions de services à la demande lors de l'exécution,

puisque dans d'autres situations, la composition est essentiellement manuelle. L'utilisateur devrait passer normalement beaucoup plus de temps pour effectuer une composition manuelle de services.

La génération d'un schéma de composition par les approches classiques peut être dans certains cas plus rapide. Mais à notre connaissance, une régénération du schéma de composition est nécessaire si les situations contextuelles changent. Avec notre approche, nous évitons le plus possible la régénération en tenant compte le plus possible des situations contextuelles lors de la génération du schéma de composition. L'utilisation des services abstraits permet par ailleurs un gain de temps par rapport à la phase de génération du plan d'exécution. Ceci est dû au fait que le *Grounding* est mis à jour quand des services concrets plus performants sont développés.

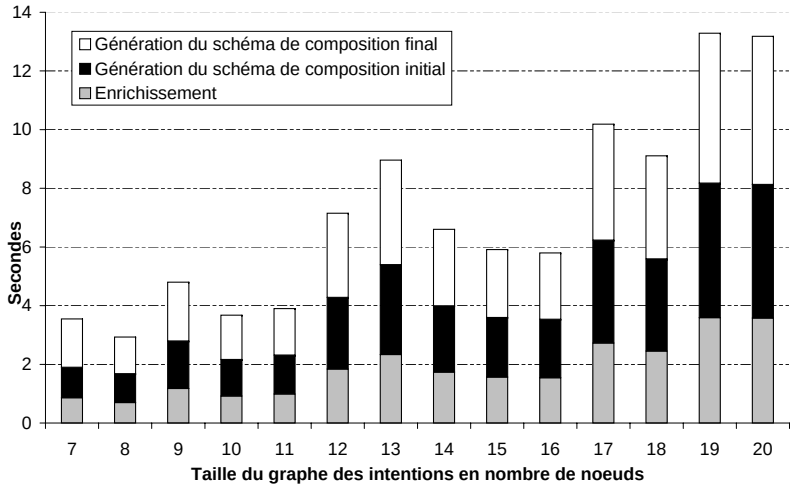

Figure 4.8 — Temps d'exécution de l'environnement de composition

4.5.2 Qualité des résultats

Dans le but d'évaluer la qualité des résultats, nous avons étudié la *précision* et le *rappel* [Salfner *et al.*, 2010] pour voir la qualité de la démarche de construction du schéma de composition initial. Ces deux métriques sont définies par deux ensembles : l'ensemble des schémas de composition trouvés et l'ensemble des schémas de composition pertinents. La métrique *précision* indique la capacité du processus de composition de retrouver uniquement les schémas de composition pertinents sans tenir compte des faux-positifs. La métrique *rappel* mesure la capacité du processus de composition à retrouver tous les schémas de composition pertinents.

La métrique *précision* est définie par l'équation suivante :

$$\frac{\{\text{Schémas de composition pertinents}\} \cap \{\text{Schémas de composition trouvés}\}}{\{\text{Schémas de composition trouvés}\}}$$

La métrique *rappel* est définie par l'équation suivante :

$$\frac{\{\text{Schémas de composition pertinents}\} \cap \{\text{Schémas de composition trouvés}\}}{\{\text{Schémas de composition pertinents}\}}$$

Afin d'évaluer ces deux métriques, nous avons défini pour chaque intention du graphe des intentions du cas d'étude 15 services abstraits aux maximum. Nous avons donc 7 intentions $I_1..I_7$ avec 15 services abstraits au maximum pour chaque intention.

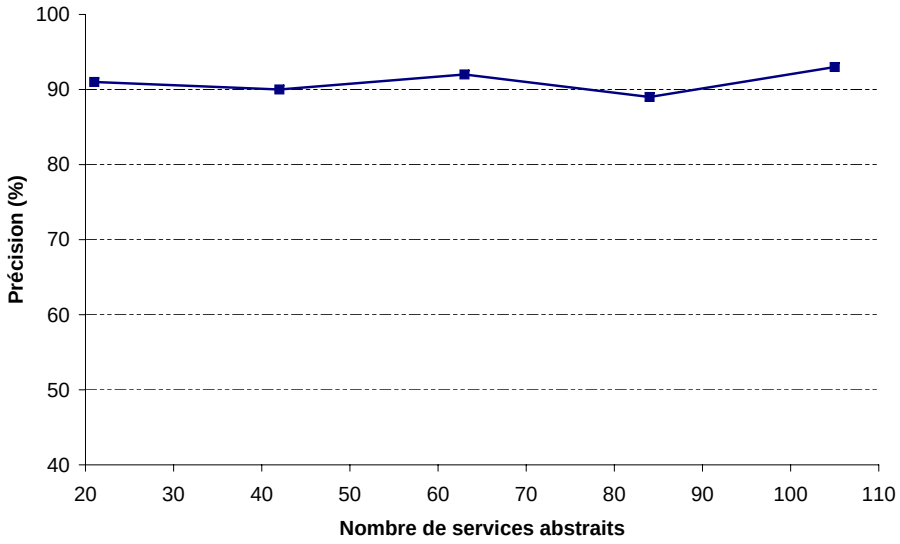

Figure 4.9 — Mesure de la métrique *précision*

Pour avoir une évaluation plus fiable de la *précision* et du *rappel*, nous avons répété les expérimentations jusqu'à réduction de l'erreur en tenant compte des recommandations définies dans [ASME, 2005]. Nous avons effectué les expérimentations avec un seuil de matching de 1,5.

Dans ce scénario du cas d'étude, l'administrateur du bâtiment souhaite réduire la consommation d'énergie en utilisant uniquement les équipements adéquats. Les résultats présentés dans la figure 4.9 indiquent que notre approche pour la génération du schéma de composition initial présente un niveau de *précision* qui vaut en moyenne 91%. Ce résultat indique que le mécanise de composition de services a une grande chance d'obtenir le schéma de composition le plus approprié pour l'utilisateur. Ceci est dû au choix d'un seuil de matching assez élevé. Le bon résultat obtenu pour la *précision* est conforté par le résultat

concernant le *rappel*, comme le montre la figure 4.10. Effectivement, le taux de *rappel* moyen pour le mécanisme de composition est proche de 89%.

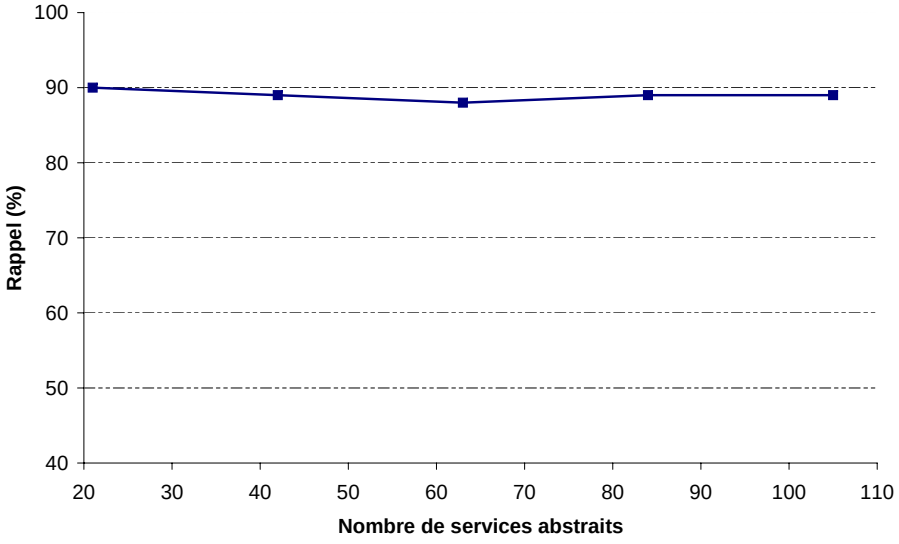

Figure 4.10 — Mesure de la métrique *rappel*

L'analyse combinée des métriques *précision* et *rappel* montre que le mécanisme de composition réussit a générer un nombre important de compositions de services qui correspondent aux besoins de l'utilisateur et cela avec un taux faible de faux-positifs. Nous pensons que le mécanisme de composition proposé permet dans les faits de générer les compositions qui correspondent au mieux aux besoins de l'utilisateur, et ceci grâce à l'utilisation du matching sémantique entre intention et service abstrait d'une part, et grâce au recours à l'affinité sémantique entre services.

Une étude de la *précision* et du *rappel* pour évaluer la qualité des résultats de la construction du schéma de composition final est nécessaire. Une telle étude permet de compléter l'étude déjà présentée et déterminer l'effet du raffinement du schéma de composition initial sur la qualité des résultats.

4.6 Conclusion

Dans ce chapitre, nous avons décrit la mise en œuvre de l'approche de composition proposée. Nous avons présenté les différents modules qui constituent notre environnement de composition, ainsi que les outils et langages utilisés pour son implémentation. Nous avons également présenté une étude de cas *(Smart Building)* qui nous a permis de valider les différentes étapes incluses dans le processus de composition. Cette étude de cas nous a permis

d'effectuer des analyses expérimentales qui ont validé nos contributions théoriques. Actuellement, nos efforts de développement se portent sur le module de génération du plan d'exécution ainsi que l'amélioration de tout l'environnement de composition. Le but étant de fournir une plateforme intégrant toutes les fonctionnalités nécessaires pour achever les étapes de composition. Cela nous permettra également de mener des tests plus approfondis. Nous étudierons la *précision* et au *rappel* pour mesurer la qualité des compositions finales fournies à l'utilisateur.

Conclusion générale et perspectives

LE développement rapide des systèmes d'information distribués et la diffusion de l'accès à Internet ont entraîné le développement de nouveaux paradigmes d'interaction entre applications. Ainsi, les paradigmes à base de composants ont évolué vers le paradigme orienté service qui s'articule autour du concept du service Web. La caractéristique importante des services Web est leur propriété de composabilité qui permet de créer des services plus complexes en combinant des services plus élémentaires en tenant compte des besoins des utilisateurs. La composition de services impliquant la capacité de sélectionner, de coordonner, d'interagir, et de faire interopérer des services Web existants constitue une tâche complexe. Cette complexité est renforcée quand il s'agit d'intégrer dynamiquement des services à la demande, et de les composer automatiquement pour répondre à des exigences qui ne sont pas réalisées par les services existants. Durant cette thèse, nous avons porté notre attention à cette problématique. En effet, nous pensons qu'une approche pour la composition de services doit offrir le potentiel de réaliser des applications flexibles et adaptables, en sélectionnant et en combinant les services de manière appropriée sur la base de la requête et du contexte de l'utilisateur.

Dans ce travail, nous avons étudié la composition des services et les approches de composition automatique. Notre étude nous a permis d'identifier les points d'intérêt de la composition et de mettre en évidence la complexité inhérente. Dans notre travail, nous avons tiré profit des avantages des stratégies proposées dans la littérature. Nous avons proposé une approche et un environnement de composition de services où la génération du schéma de composition est faite en partie au moment de l'exécution. Cette approche repose sur des éléments fournis au moment de la conception qui permettent une certaine flexibilité sans avoir besoin de construire une composition de services à partir de zéro au moment de l'exécution. Dans ce travail, nous avons recours aux intentions qui permettent de structurer les besoins des utilisateurs. Nous avons défini le modèle des intentions en utilisant une structure sémantique basée sur OWL qui nous a permis d'ajouter de nouveaux concepts et propriétés. Nous avons eu par ailleurs recours à des services abstraits que nous avons définis en se basant sur le langage OWL-S. Notre approche se compose principalement de quatre étapes.

La première étape prend en entrée un graphe des intentions structuré sémantiquement. Ce graphe explicite les relations entre les besoins de l'utilisateur. Des règles sémantiques que nous avons définies enrichissent le graphe et explicitent les relations implicites. dans la seconde étape, le graphe des intentions est parcouru pour identifier les services abstraits nécessaires. Le matching sémantique est utilisé pour la sélection des services abstraits. Nous avons défini des règles qui permettent de choisir les services abstraits les plus adéquats aux intentions. Par ailleurs, pour établir le flux de contrôle entre les services abstraits, nous nous sommes basés sur les relations entre les intentions. Pour s'assurer que les services abstraits sont connectés d'une manière optimale, nous avons calculé l'affinité sémantique entre les services. Ce calcul permet d'établir le flux de données. A la fin de cette étape, nous obtenons le schéma de composition initial. La troisième étape consiste à générer le schéma de composition final à l'aide d'un mécanisme de raffinement des services abstraits en utilisant des techniques de matching sémantique. En fait, la partie processus du service abstrait est parcourue et les différents sous-processus sont raffinés en étant mis en correspondance avec des services abstraits. La prise en compte du contexte permet de choisir les services abstraits qui tiennent compte des éléments du contexte précisés dans le graphe des intentions. Le raisonnement sémantique pour le matching et pour le raffinement vérifie que les services sélectionnés prennent en considération les éléments du contexte. Dans la quatrième étape, le plan d'exécution est généré à partir du schéma de composition final. Pour chaque service abstrait atomique, l'ensemble des services concrets qui lui sont associés sont parcourus. En tenant compte des contraintes non fonctionnelles exprimées dans l'intention, le service concret adéquat est sélectionné. Dans notre travail, nous avons proposé une architecture de l'environnement de composition qui permet de mettre en oeuvre les quatre étapes précédemment citées. Pour illustrer notre travail, nous nous sommes intéressés au cas d'étude du bâtiment intelligent. Le but principal du gestionnaire du bâtiment intelligent est d'assurer le confort des utilisateurs tout en optimisant la consommation de l'énergie. Nous avons modélisé les besoins à travers un graphe des intentions qui a été traité par notre environnement de composition. Nous avons généré un des ensembles de graphes des intentions de différentes tailles pour les traiter et générer les schémas de composition finaux correspondants. Ceci nous a permis d'observer les temps d'exécution en fonction de la taille des graphes des intentions et la qualité des résultats en utilisant les métriques précision et rappel.

Le travail que nous avons présenté dans ce mémoire a pu donner des réponses à certains points problématiques que nous avons évoqués, mais a aussi dégagé un ensemble de questions et de perspectives.

Pour pouvoir être plus complet, il serait intéressant d'étudier la possibilité d'intégrer un

module de reconnaissance des intentions à partir de travaux existants. Ceci permettrait de simplifier la tâche de l'utilisateur ou/et du bénéficiaire de la composition. Par ailleurs, traiter le changement des intentions au cours de l'exécution ouvre des perspectives importantes par rapport au déroulement du processus de génération des schémas de composition. L'idée serait de permettre des changements suivant des niveaux de criticité pour ne pas tomber dans des systèmes instables par le fait des changements rapides.

La prise en compte du contexte ne s'appuie pas pour le moment sur une structure sémantique élaborée. Des règles de raisonnement sur les éléments du contexte peuvent améliorer la sélection des services abstraits. L'idée consiste à étudier la possibilité de déduire de nouvelles connaissances à partir du modèle de contexte. Il serait intéressant d'explorer plus les possibilités d'interprétation des informations représentées dans un modèle. Un tel processus serait particulièrement intéressant pour l'extraction de connaissances à partir d'un ensemble d'informations incomplètes ou ambiguës.

L'étape de génération du plan d'exécution présente des défis importants comme la propagation des contraintes non fonctionnelles et la volatilité des services concrets. En effet, la sélection des services concrets doit minimiser l'impact des ajouts et des retraits de services par des fournisseurs de services. La minimisation du temps d'exécution du processus de composition est nécessaire. Ceci peut se faire en utilisant la classification des ontologies.

Enfin, l'association d'un aspect temporel aux profils des service abstraits est une perspective intéressante pour ce travail. L'idée ici serait de permettre la définition de contraintes temporelles dans les profils. Chaque profil aurait ainsi des contraintes qui devraient être respecter pendant « l'exécution » du service et il ne serait plus valide ni applicable si les contraintes ne sont pas vérifiées. Cette perspective rendrait possible, par exemple, la définition d'une politique de contrôle d'accès valable pour une période donnée.

Liste des publications

Emna FKI, Saïd TAZI et Mohamed JMAIEL : A semantic driven approach for an automated composition based on abstract services. Dans IEEE 10th International Conference on e-Business Engineering, ICEBE, Beijin, Chine, October 23 – 25, 2015, 2015. to appear.

Emna FKI, Mohamed JMAIEL, Chantal SOULÉ-DUPUY et Saïd TAZI : A flexible approach for service composition using service patterns. Dans the 27th Symposium On Applied Computing, pages 1526 – 1534, Trento, Italy, 2012.

Emna FKI, Chantal SOULÉ-DUPUY, Saïd TAZI et Mohamed JMAIEL : Vers une composition de services basée sur les patrons de services et dirigée par les intentions. Dans 5ème Conférence francophone sur les Architectures Logicielles, pages 29 – 42, Lille (France), 2011.

Emna FKI, Chantal SOULÉ-DUPUY, Saïd TAZI et Mohamed JMAIEL : Intention driven service composition with patterns. Dans the 12th International Conference on Enterprise Information Systems, pages 236 – 241, Madeira, Portugal, 2010.

Emna FKI, Saïd TAZI et Chantal SOULÉ-DUPUY : Towards a user intention aware service composition. Dans the 10th Annual International Conference on New Technologies of Distributed Systems, pages 113 – 120, Touzeur, Tunisia, 2010.

Bibliographie

Vikas AGARWAL, Koustuv DASGUPTA, Neeran KARNIK, Arun KUMAR, Ashish KUNDU, Sumit MITTAL et Biplav SRIVASTAVA : A service creation environment based on end to end composition of web services. *Dans Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 128–137, New York, NY, USA, 2005. ACM. ISBN 1-59593-046-9. URL `http://doi.acm.org/10.1145/1060745.1060768`.

Rohit AGGARWAL, Kunal VERMA, John MILLER et William MILNOR : Constraint driven web service composition in meteor-s. *Dans SCC '04 : Proceedings of the 2004 IEEE International Conference on Services Computing*, pages 23–30, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2225-4.

Assel AKZHALOVA et Iman POERNOMO : Model driven approach for dynamic service composition based on qos constraints. *2014 IEEE World Congress on Services*, 0:590–597, 2010.

Mohammad ALRIFAI, Dimitrios SKOUTAS et Thomas RISSE : Selecting skyline services for qos-based web service composition. *Dans Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 11–20. ACM, 2010. ISBN 978-1-60558-799-8.

Ismailcem BUDAK ARPINAR, Ruoyan ZHANG, Boanerges ALEMAN-MEZA et Angela MADUKO : Ontology-driven web services composition platform. *Inf. Syst. E-Business Management*, 3(2):175–199, 2005.

A ARSANJANI : Service-oriented modeling and architecture - how to identify, specify and realize services for your soa. *Dans Technical article, SOA and Web Services Center of Excellence, IBM, Software Group (9 November 2004). Online : http ://www.ibm.com/developerworks/webservices/library/ws-soa-design1*, 2004.

ASME : The Performance Test Standard PTC 19.1-2005 Test Uncertainty. Rapport technique, American Society of Mechanical Engineers, NewYork,États-Unis, 2005.

Lerina AVERSANO, Gerardo CANFORA et Anna CIAMPI : An algorithm for web service discovery through their composition. *2013 IEEE 20th International Conference on Web Services*, 0:332, 2004.

Fayçal BACHTARZI : *Une Approche de Composition des Services Web Basée Transformation de graphes*. Thèse de doctorat, Université Abdelhamid Mehri Constantine 2, 2014.

Peter BARTALOS et Mária BIELIKOVÁ : Automatic dynamic web service composition : A survey and problem formalization. *Comput. and Inf*, pages 793–827, 2011.

Khalid BELHAJJAME, Suzanne M. EMBURY, Norman W. PATON, Robert STEVENS et Carole A. GOBLE : Automatic annotation of web services based on workflow definitions. *ACM Trans. Web*, 2(2):11 :1–11 :34, mai 2008. ISSN 1559-1131. URL `http://doi.acm.org/10.1145/1346237.1346239`.

Boualem BENATALLAH, Quan Z. SHENG et Marlon DUMAS : The self-serv environment for web services composition. *IEEE Internet Computing*, 7(1):40–48, 2003.

Michael E. BRATMAN : *Intention, Plans, and Practical Reason*. Cambridge University Press, March 1999. ISBN 1575861925. URL `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/1575861925`.

Peter F BROWN, Rebekah METZ et Booz Allen HAMILTON : Reference model for service oriented architecture 1.0, 2006.

Antonio BUCCHIARONE et Stefania GNESI : A Survey on Services Composition Languages and Models. *Dans Proceedings of International Workshop on Web Services Modeling and Testing 2006 (WS-MaTe 2006)*, 2006.

Gerardo CANFORA, Massimiliano DI PENTA, Raffaele ESPOSITO et Maria Luisa VILLANI : An approach for qos-aware service composition based on genetic algorithms. *Dans Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, pages 1069–1075. ACM, 2005a. ISBN 1-59593-010-8.

Gerardo CANFORA, Massimiliano DI PENTA, Raffaele ESPOSITO et Maria Luisa VILLANI : An approach for qos-aware service composition based on genetic algorithms. *Dans Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, GECCO '05, pages 1069–1075, New York, NY, USA, 2005b. ACM. ISBN 1-59593-010-8. URL `http://doi.acm.org/10.1145/1068009.1068189`.

Fabio CASATI, Malú CASTELLANOS, Umeshwar DAYAL et Ming-Chien SHAN : Probabilistic, context-sensitive, and goal-oriented service selection. *Dans* Marco AIELLO, Mikio AOYAMA, Francisco CURBERA et Mike P. PAPAZOGLOU, éditeurs : *ICSOC*, pages 316–321. ACM, 2004. ISBN 1-58113-871-7.

Fabio CASATI, Ski ILNICKI, Li jie JIN, Vasudev KRISHNAMOORTHY et Ming-Chien SHAN : Adaptive and dynamic service composition in *flow*. *Dans CAiSE*, pages 13–31, 2000.

Corine CAUVET et Gwladys GUZELIAN : Business process modeling : A service-oriented approach. *Dans 41st Hawaii International International Conference on Systems Science (HICSS-41 2008), Proceedings, 7-10 January 2008, Waikoloa, Big Island, HI, USA*, page 98, 2008. URL http://dx.doi.org/10.1109/HICSS.2008.84.

Pat Pik-Wah CHAN et Michael R. LYU : Dynamic web service composition : A new approach in building reliable web service. *Dans AINA*, pages 20–25. IEEE Computer Society, 2008.

Carl K. CHANG, Hsin yi JIANG, Hua MING et Katsunori OYAMA : Situ : A situation-theoretic approach to context-aware service evolution. *IEEE Transactions on Services Computing*, 99 (PrePrints):261–275, 2009. ISSN 1939-1374.

Annie CHEN : Context-aware collaborative filtering system : Predicting the user's preference in the ubiquitous computing environment. *Dans* Thomas STRANG et Claudia LINNHOFF-POPIEN, éditeurs : *LoCA*, volume 3479 de *Lecture Notes in Computer Science*, pages 244–253. Springer, 2005. ISBN 3-540-25896-5.

Harry CHEN, Tim FININ et Anupam JOSHI : An ontology for context-aware pervasive computing environments. *Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review*, 18:197–207, 2003a.

Kun CHEN, Jiuyun XU et Stephan REIFF-MARGANIEC : Markov-htn planning approach to enhance flexibility of automatic web service composition. *Dans ICWS*, pages 9–16. IEEE, 2009. URL http://dblp.uni-trier.de/db/conf/icws/icws2009.html#ChenXR09.

Liming CHEN, Nigel SHADBOLT, Carole GOBLE, Feng TAO, Simon COX, Colin PULESTON et Paul R. SMART : Towards a knowledge-based approach to semantic service composition. *Dans Second International Semantic Web Conference (ISWC2003)*, 2003b.

Erik CHRISTENSEN, Francisco CURBERA, Greg MEREDITH et Sanjiva WEERAWARANA : Web services description language (WSDL) 1.1. W3c note, World Wide Web Consortium, March 2001. URL http://www.w3.org/TR/wsdl.

Pierre CHÂTEL, Jacques MALENFANT et Isis TRUCK : Qos-based late-binding of service invocations in adaptive business processes. *Dans ICWS*, pages 227–234. IEEE Computer Society, 2010. ISBN 978-0-7695-4128-0.

Soon Ae CHUN, Vijayalakshmi ATLURI et Nabil R. ADAM : Using semantics for policy-based web service composition. *Distrib. Parallel Databases*, 18(1):37–64, 2005. ISSN 0926-8782.

Philip R. COHEN et Hector J. LEVESQUE : Intention is choice with commitment. *Artificial Intelligence*, 42(2-3):213–261, 1990.

OASIS COMMITTEE : UDDI Spec Technical Committee Draft 3.0.2. OASIS Committee Draft, 2004.

Patrícia Dockhorn COSTA, Luís Ferreira PIRES, Marten van SINDEREN et Diego RIOS : Towards a services platform for context-aware applications. Rapport technique, University of Twente, 2003.

Anind K. DEY et Gregory D. ABOWD : Towards a better understanding of context and context-awareness. *Dans Workshop on The What, Who, Where, When, and How of Context-Awareness*, 2000 Conference on Human Factors in Computing Systems, 2000.

Anind K. DEY, Daniel SALBER, Masayasu FUTAKAWA et Gregory D. ABOWD : An architecture to support context-aware applications. Technical report, Georgia Institute of Technology, 1999.

Maja D'HONDT et Viviane JONCKERS : Hybrid aspects for weaving object-oriented functionality and rule-based knowledge. *Dans* Gail C. MURPHY et Karl J. LIEBERHERR, éditeurs : *AOSD*, pages 132–140. ACM, 2004. ISBN 1-58113-842-3.

Michael DIBERNARDO, Rachel POTTINGER et Mark WILKINSON : Semi-automatic web service composition for the life sciences using the biomoby semantic web framework. *Journal of Biomedical Informatics*, 41(5):837–847, 2008.

Demian Antony D'MELLO et V. S. ANANTHANARAYANA : Dynamic web service composition based on operation flow semantics. *Dans Information Systems, Technology and Management - 4th International Conference, ICISTM 2010, Bangkok, Thailand, March 11-13, 2010. Proceedings*, pages 111–122, 2010.

Mahesh H. DODANI : From objects to services : A journey in search of component reuse nirvana. *Journal of Object Technology*, 3(8):49–54, 2004. URL `http://dblp.uni-trier.de/db/journals/jot/jot3.html#Dodani04c`.

Schahram DUSTDAR et Mike P. PAPAZOGLOU : Services and service composition - an introduction. *it - Information Technology*, 50(2):86–92, 2008a.

Schahram DUSTDAR et Mike P. PAPAZOGLOU : Services and service composition - an introduction (services und service komposition - eine einführung). *it - Information Technology*, 50(2):86–92, 2008b. URL `http://dx.doi.org/10.1524/itit.2008.0468`.

Kutluhan EROL, James HENDLER et Dana S. NAU : Semantics for hierarchical task-network planning. Rapport technique, ARPA Rome Laboratory, 1995.

Dan Wu EVREN, Dan WU, Evren SIRIN, James HENDLER, Dana NAU et Bijan PARSIA : Automatic web services composition using shop2. *Dans In Workshop on Planning for Web Services*, 2003.

Yong-Yi FANJIANG et Yang SYU : Semantic-based automatic service composition with functional and non-functional requirements in design time. *Inf. Softw. Technol.*, 56(3):352–373, 2014. ISSN 0950-5849.

Matthias FLÜGGE et Diana TOURTCHANINOVA : Ontology-derived activity components for composing travel web services. *Dans* Robert TOLKSDORF et Rainer ECKSTEIN, éditeurs : *Berliner XML Tage*, pages 133–150. XML-Clearinghouse, 2004.

Howard FOSTER : Behaviour analysis and verification of web service compositions, 2004.

Keita FUJII et Tatsuya SUDA : Semantics-based context-aware dynamic service composition. *ACM Trans. Auton. Adapt. Syst.*, 4(2):12 :1–12 :31, mai 2009. ISSN 1556-4665. URL `http://doi.acm.org/10.1145/1516533.1516536`.

Sahin Cem GEYIK, Boleslaw K. SZYMANSKI et Petros ZERFOS : Robust dynamic service composition in sensor networks. *IEEE Trans. Serv. Comput.*, 6(4):560–572, octobre 2013. ISSN 1939-1374.

Carlos GRANELL, Laura DÍAZ et Michael GOULD : Service-oriented applications for environmental models : Reusable geospatial services. *Environ. Model. Softw.*, 25(2):182–198, février 2010. ISSN 1364-8152.

Christin GROBA et Siobhán CLARKE : Opportunistic service composition in dynamic ad hoc environments. volume 7, pages 642–653, 2014.

Barbara GROSZ et Candy SIDNER : Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3), 1986.

Thomas R. GRUBER : A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220, juin 1993. ISSN 1042-8143. URL http://dx.doi.org/10.1006/knac.1993.1008.

Ikbel GUIDARA, Nawal GUERMOUCHE, Tarak CHAARI, Saïd TAZI et Mohamed JMAIEL : Pruning based service selection approach under qos and temporal constraints. *Dans 2014 IEEE International Conference on Web Services, ICWS, 2014, Anchorage, AK, USA, June 27 - July 2, 2014*, pages 9–16, 2014.

Joyce El HADDAD, Maude MANOUVRIER et Marta RUKOZ : Tqos : Transactional and qos-aware selection algorithm for automatic web service composition. *IEEE Transactions on Services Computing*, 3(1):73–85, 2010.

Yanbo HAN, Jing WANG et Peng ZHANG : Business-oriented service modeling : A case study. *Simulation Modelling Practice and Theory*, 17(8):1413–1429, 2009. URL http://dx.doi.org/10.1016/j.simpat.2009.06.011.

Seyyed Vahid HASHEMIAN et Farhad MAVADDAT : A graph-based framework for composition of stateless web services. *Dans ECOWS*, pages 75–86. IEEE Computer Society, 2006. ISBN 0-7695-2737-X.

Ourania HATZI, Dimitris VRAKAS, Mara NIKOLAIDOU, Nick BASSILIADES, Dimosthenis ANAGNOSTOPOULOS et Ioannis P. VLAHAVAS : An integrated approach to automated semantic web service composition through planning. *IEEE Transactions on Services Computing*, 5(3):319–332, 2012.

Liang HONG, Zongyi HE, Jian XIANG et Shixiang LI : Geography information web service semantic description and automatic discovery based on web service and OWL-S. *Dans First International Workshop on Database Technology and Applications, DBTA 2009, Wuhan, Hubei, China, April 25-26, 2009, Proceedings*, pages 517–521, 2009.

Michael N. HUHNS et Munindar P. SINGH : Service-oriented computing : Key concepts and principles. *IEEE Internet Computing*, 9(1):75–81, 2005.

Jing JIANG et Quan BAI : Correlated contribution analysis for service composition in dynamic environments. *Dans 2013 IEEE International Conference on Services Computing, Santa Clara, CA, USA, June 28 - July 3, 2013*, pages 113–119, 2013.

Graham KLYNE et Jeremy J. CARROLL : Resource description framework (RDF) : Concepts and abstract syntax. World Wide Web Consortium, Recommendation REC-rdf-concepts-20040210, February 2004.

Holger KNUBLAUCH, Ray W. FERGERSON, Natalya Fridman NOY et Mark A. MUSEN : The protégé OWL plugin : An open development environment for semantic web applications. *Dans The Semantic Web - ISWC 2004 : Third International Semantic Web Conference,Hiroshima, Japan, November 7-11, 2004. Proceedings*, pages 229–243, 2004. URL http://dx.doi.org/10.1007/978-3-540-30475-3_17.

S. KONA, A BANSAL et G. GUPTA : Automatic composition of semantic web services. *Dans Web Services, 2007. ICWS 2007. IEEE International Conference on*, pages 150–158, July 2007.

Jacek KOPECKÝ, Tomas VITVAR, Carine BOURNEZ et Joel FARRELL : Sawsdl : Semantic annotations for wsdl and xml schema. *IEEE Internet Computing*, 11(6):60–67, 2007. URL http://dx.doi.org/10.1109/MIC.2007.134.

Soufiene LAJMI, Chirine GHEDIRA et Khaled GHÉDIRA : Cbr method for web service composition. *Dans SITIS*, pages 314–326, 2006.

Freddy LÉCUÉ et Alain LÉGER : A formal model for semantic web service composition. *Dans International Semantic Web Conference*, volume 4273 de *Lecture Notes in Computer Science*, pages 385–398. Springer, 2006. ISBN 3-540-49029-9. URL http://dblp.uni-trier.de/db/conf/semweb/iswc2006.html#LecueL06.

Freddy LÉCUÉ et Nikolay MEHANDJIEV : Towards scalability of quality driven semantic web service composition. *Dans ICWS*, pages 469–476. IEEE, 2009.

Helan LIANG, Yanhua DU et Sujian LI : An improved genetic algorithm for service selection under temporal constraints in cloud computing. *Dans* Xuemin LIN, Yannis MANOLOPOULOS, Divesh SRIVASTAVA et Guangyan HUANG, éditeurs : *WISE (2)*, volume 8181 de *Lecture Notes in Computer Science*, pages 309–318. Springer, 2013. ISBN 978-3-642-41153-3. URL http://dblp.uni-trier.de/db/conf/wise/wise2013-2.html#LiangDL13.

Qianhui LIANG, Peipei LI, Patrick C.K. HUNG et Xindong WU : Clustering web services for automatic categorization. *2013 IEEE International Conference on Services Computing*, 0:380–387, 2009.

Donghui LIN, Chunqi SHI et Toru ISHIDA : Dynamic service selection based on context-aware qos. *Dans IEEE SCC*, pages 641–648. IEEE Computer Society, 2012.

Manshan LIN, Heqing GUO et Jianfei YIN : Goal description language for semantic web service automatic composition. *Dans SAINT*, pages 190–196, 2005.

Naiwen LIN, Ugur KUTER et Evren SIRIN : Web service composition with user preferences. *Dans The Semantic Web : Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008, Proceedings*, pages 629–643, 2008. URL http://dx.doi.org/10.1007/978-3-540-68234-9_46.

Shao Yuan LU, Kuo-Hsun HSU et Li-Jing KUO : A semantic service match approach based on wordnet and swrl rules. *Dans IEEE 10th International Conference on e-Business Engineering (ICEBE)*, pages 419–422, 2013.

Nebil Ben MABROUK, Nikolaos GEORGANTAS et Valérie ISSARNY : Set-based bi-level optimisation for qos-aware service composition in ubiquitous environments. *Dans 2015 IEEE International Conference on Web Services, ICWS 2015, New York, NY, USA, June 27 - July 2, 2015*, pages 25–32, 2015.

David MARTIN, Mark BURSTEIN, Jerry HOBBS, Ora LASSILA, Drew MCDERMOTT, Sheila MCILRAITH, Srini NARAYANAN, Massimo PAOLUCCI, Bijan PARSIA, Terry PAYNE, Evren SIRIN, Naveen SRINIVASAN et Katia SYCARA : *OWL-S : Semantic Markup for Web Services*, 2004.

Deborah L. MCGUINNESS et Frank van HARMELEN : Owl web ontology language overview. Rapport technique REC-owl-features-20040210, W3C, 2004.

Brahim MEDJAHED, Athman BOUGUETTAYA et Ahmed K. ELMAGARMID : Composing web services on the semantic web. *The VLDB Journal*, 12(4):333–351, November 2003. ISSN 1066-8888. URL http://dx.doi.org/10.1007/s00778-003-0101-5.

Daniel A. MENASCÉ, Emiliano CASALICCHIO et Vinod K. DUBEY : A heuristic approach to optimal service selection in service oriented architectures. *Dans* Alberto AVRITZER, Elaine J. WEYUKER et C. Murray WOODSIDE, éditeurs : *WOSP*, pages 13–24. ACM, 2008. ISBN 978-1-59593-873-2. URL http://dblp.uni-trier.de/db/conf/wosp/wosp2008.html#MenasceCD08.

Hua MING, Katsunori OYAMA et Carl K. CHANG : Human-intention driven self adaptive software evolvability in distributed service environments. *Future Trends of Distributed Computing Systems, IEEE International Workshop*, 0:51–57, 2008. ISSN 1071-0485.

Nilo MITRA et Yves LAFON : Soap version 1.2 part 0 : Primer (second edition). World Wide Web Consortium, Recommendation REC-soap12-part0-20070427, April 2007.

F. MOHR, A. JUNGMANN et H. Kleine BÜNING : Automated online service composition. *Dans Proceedings of the 12th IEEE International Conference on Services Computing (SCC)*, 2015.

Sonia Ben MOKHTAR, Damien FOURNIER, Nikolaos GEORGANTAS, Valérie ISSARNY, Sonia. Ben MOKHTAR, Damien. FOURNIER et Nikolaos. GEORGANTAS : Context-aware service composition in pervasive computing environments. *Dans In RISE,volume3943ofLNCS,pages 129 144*. Springer, 2005.

Vito MORREALE, Susanna BONURA, Giuseppe FRANCAVIGLIA, Fabio CENTINEO, Massimo COSSENTINO et Salvatore GAGLIO : Reasoning about goals in bdi agents : the practionist framework. *Dans In Proceedings of Joint Workshop From Objects to Agents*, 2006.

Pathathai NA-LUMPOON, Marie-Christine FAUVET et Ahmed LBATH : Toward a framework for automated service composition and execution. *Dans 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*, pages 1–8, 2014.

Robert NECHES, Richard FIKES, Tim FININ, Tom GRUBER, Ramesh PATIL, Ted SENATOR et William R. SWARTOUT : Enabling technology for knowledge sharing. *AI Mag.*, 12(3):36–56, septembre 1991. ISSN 0738-4602. URL `http://dl.acm.org/citation.cfm?id=123768.123775`.

OASIS WEB SERVICES BUSINESS PROCESS EXECUTION LANGUAGE (WSBPEL) TECHNICAL COMMITTEE : Web Service Business Process Execution Language Version 2.0, 2007.

Massimo PAOLUCCI, Takahiro KAWAMURA, Terry R. PAYNE et Katia P. SYCARA : Semantic matching of web services capabilities. *Dans International Semantic Web Conference*, pages 333–347, 2002.

Michael P. PAPAZOGLOU, Paolo TRAVERSO, Schahram DUSTDAR et Frank LEYMANN : Service-Oriented Computing : A Research Roadmap. *International Journal of Cooperative Information Systems (IJCIS)*, 17(2):233–255, June 2008.

Jason PASCOE : Adding generic contextual capabilities to wearable computers. *Dans Proceedings of the 2Nd IEEE International Symposium on Wearable Computers*, ISWC'98, pages 92–, Washington, DC, USA, 1998. IEEE Computer Society. ISBN 0-8186-9074-7. URL `http://dl.acm.org/citation.cfm?id=857199.858020`.

Terry R. PAYNE et Ora LASSILA : Guest editors' introduction : Semantic web services. *IEEE Intelligent Systems*, pages 14–15, 2004.

Joachim PEER : A pddl based tool for automatic web service composition. *Dans In Proceedings of the Second Intl Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR*, pages 149–163. Springer Verlag, 2004.

Hui PENG : Context aware semantic web services description and match. *Dans 2013 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC 2013, Beijing, China, October 10-12, 2013*, pages 210–213. IEEE Computer Society, 2013.

Pierluigi PLEBANI et Barbara PERNICI : Urbe : Web service retrieval based on similarity evaluation. *IEEE Transactions on Knowledge and Data Engineering*, 21(11):1629–1642, 2009. ISSN 1041-4347.

Shankar R. PONNEKANTI et Armando FOX : Sword : A developer toolkit for web service composition. *Dans Proceedings of the 11th International WWW Conference (WWW2002)*, Honolulu, HI, USA, 2002.

Roy RADA, Hafedh MILI, Ellen BICKNELL et Maria BLETTNER : Development and application of a metric on semantic nets. *Dans IEEE Transactions on Systems, Man and Cybernetics*, pages 17–30, 1989.

René RAMACHER et Lars MÖNCH : Service selection with runtime aspects : A hierarchical approach. *IEEE Transactions on Services Computing*, 8(3):481–493, 2015.

Anand S. RAO et Michael P. GEORGEFF : Modeling rational agents within a bdi-architecture. *Dans Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 473–484, 1991.

Jinghai RAO et Peep KÜNGAS : Logic-based web services composition : From service description to process model. *Dans In Intl. Conference on Web Services (ICWS*, pages 446–453. IEEE, 2004.

Kaijun REN, Nong XIAO et Jinjun CHEN : Building quick service query list using wordnet and multiple heterogeneous ontologies toward more realistic service composition. *IEEE Transactions on Services Computing*, 4(3):216–229, 2011. ISSN 1939-1374.

Colette ROLLAND, Manuele KIRSCH-PINHEIRO et Carine SOUVEYET : An intentional approach to service engineering. *IEEE T. Services Computing*, 3(4):292–305, 2010. URL `http://dblp.uni-trier.de/db/journals/tsc/tsc3.html#RollandKS10`.

Felix SALFNER, Maren LENK et Miroslaw MALEK : A survey of online failure prediction methods. *ACM Comput. Surv.*, 42(3):10 :1–10 :42, mars 2010. ISSN 0360-0300.

Ivo José Garcia Dos SANTOS, Matthias FLÜGGE, Neil Paiva TIZZO et Edmundo Roberto Mauro MADEIRA : Challenges and techniques on the road to dynamically compose

web services. *Dans Proceedings of the 6th International Conference on Web Engineering, ICWE 2006, Palo Alto, California, USA*, pages 40–47. ACM, 2006.

Richard E. SCHANTZ et Douglas C. SCHMIDT : Middleware for distributed systems : Evolving the common structure for network-centric applications. *Encyclopedia of Software Engineering*, 2002.

Bill N. SCHILIT, Norman ADAMS et Roy WANT : Context-aware computing applications. *Dans Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*, WMCSA '94, pages 85–90, Washington, DC, USA, 1994. IEEE Computer Society. ISBN 978-0-7695-3451-0. URL http://dx.doi.org/10.1109/WMCSA.1994.16.

Bill N. SCHILIT et Marvin M. THEIMER : Disseminating active map information to mobile hosts. *Netwrk. Mag. of Global Internetwkg.*, 8(5):22–32, septembre 1994. ISSN 0890-8044. URL http://dx.doi.org/10.1109/65.313011.

Ehsan SHARIFI, Reza A. MOGHADAM, Fernando BOBILLO et Mohamad M. EBADZADEH : A fuzzy framework for semantic web service description, matchmaking, ranking and selection. *Dans Eighth International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2011, 26-28 July 2011, Shanghai, China*, pages 621–625, 2011.

Kwang Mong SIM et Pui Tak WONG : Web-based information retrieval using agent and ontology. *Dans* Ning ZHONG, Yiyu YAO, Jiming LIU et Setsuo OHSUGA, éditeurs : *Web Intelligence*, volume 2198 de *Lecture Notes in Computer Science*, pages 384–388. Springer, 2001. ISBN 3-540-42730-9.

Evren SIRIN, James HENDLER et Bijan PARSIA : Semi-automatic composition of web services using semantic descriptions. *Dans In Web Services : Modeling, Architecture and Infrastructure workshop in ICEIS 2003*, pages 17–24, 2002.

Evren SIRIN, Bijan PARSIA, Dan WU, James HENDLER et Dana NAU : Htn planning for web service composition using shop2. *Web Semantics : Science, Services and Agents on the World Wide Web*, 1(4):377–396, October 2004. URL http://dx.doi.org/10.1016/j.websem.2004.06.005.

Tomas SKERSYS, Lina TUTKUTE et Rimantas BUTLERIS : The enrichment of bpmn business process model with sbvr business vocabulary and rules. *Journal of Computing and Information Technology*, 20(3):143–150, 2012. URL http://dblp.uni-trier.de/db/journals/cit/cit20.html#SkersysTB12.

David SPROTT et Lawrence WILKES : Understanding service-oriented architecture. *CBDI Forum*, January 2004.

Biplav SRIVASTAVA et Jana KOEHLER : Web service composition - current solutions and open problems. *Dans In : ICAPS 2003 Workshop on Planning for Web Services*, pages 28–35, 2003.

Yang SYU, Yong-Yi FANJIANG, Jong-Yih KUO et Shang-Pin MA : Towards a genetic algorithm approach to automating workflow composition for web services with transactional and qos-awareness. *Dans SERVICES*, pages 295–302. IEEE Computer Society, 2011. ISBN 978-1-4577-0879-4. URL http://dblp.uni-trier.de/db/conf/services/services2011.html#SyuFKM11.

Yang SYU, Shang-Pin MA, Jong-Yih KUO et Yong-Yi FANJIANG : A survey on automated service composition methods and related techniques. *Dans* Louise E. MOSER, Manish PARASHAR et Patrick C. K. HUNG, éditeurs : *IEEE SCC*, pages 290–297. IEEE, 2012. ISBN 978-1-4673-3049-7.

Clemens SZYPERSKI : *Component Software : Beyond Object-oriented Programming*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1998. ISBN 0-201-17888-5.

Xinhuai TANG, Feilong TANG, Liu BING et Delai CHEN : Dynamic web service composition based on service integration and htn planning. *Dans IMIS*, pages 307–312. IEEE Computer Society, 2013.

David A. TAYLOR : *Object Technology (2Nd Ed.) : A Manager's Guide*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998. ISBN 0-201-30994-7.

Ian J. TAYLOR, Matthew S. SHIELDS, Ian WANG et Roger PHILP : Distributed p2p computing within triana : A galaxy visualization test case. *Dans 17th International Parallel and Distributed Processing Symposium (IPDPS 2003), 22-26 April 2003, Nice, France, CD-ROM/Abstracts Proceedings*, page 16, 2003. URL http://dx.doi.org/10.1109/IPDPS.2003.1213094.

Said TAZI : *Explicitation des intentions et conception de documents*. Habilitation à diriger des recherches, LAAS, Jun 2005. URL http://tel.archives-ouvertes.fr/tel-00069300.

Annette Ten TEIJE, Frank van HARMELEN et Bob J. WIELINGA : Configuration of web services as parametric design. *Dans ECAI*, pages 1097–1098, 2004.

Michael TOMASELLO, Malinda CARPENTER, Josep CALL, Tanya BEHNE et Henrike MOLL : Understanding and sharing intentions : The origins of cultural cognition. *Behavioral and Brain Sciences*, 28, no. 5, pp. 675-691, 2005.

Moe T. TUT et David EDMOND : The use of patterns in service composition. *Dans International Workshop on Web Services, E-Business, and the Semantic Web*, pages 28–40, London, UK, 2002. Springer-Verlag.

Steve VINOSKI : An overview of middleware. *Dans Ada-Europe*, volume 3063 de *Lecture Notes in Computer Science*, pages 35–51. Springer, 2004. ISBN 3-540-22011-9. URL `http://dblp.uni-trier.de/db/conf/adaEurope/adaEurope2004.html#Vinoski04`.

Chien-Min WANG, Shun-Te WANG, Hsi-Min CHEN et Chi-Chang HUANG : A reliability-aware approach for web services execution planning. *Dans IEEE SCW*, pages 278–283. IEEE Computer Society, 2007.

Hongbing WANG, Qin WU, Xin CHEN, Qi YU, Zibin ZHENG et Athman BOUGUETTAYA : Adaptive and dynamic service composition via multi-agent reinforcement learning. *Dans ICWS*, pages 447–454. IEEE Computer Society, 2014.

Hongbing WANG, Xuan ZHOU, Xiang ZHOU, Weihong LIU, Wenya LI et Athman BOUGUETTAYA : Adaptive service composition based on reinforcement learning. *Dans ICSOC*, pages 92–107, 2010.

Jianping WANG : Exploiting mobility prediction for dependable service composition in wireless mobile ad hoc networks. *IEEE Transactions on Services Computing*, 4(1):44–55, 2011. ISSN 1939-1374.

Andy WARD et Alan JONES : A new location technique for the active office, 1997.

Richard W. WEYHRAUCH : Prolegomena to a theory of mechanized formal reasoning. *Artif. Intell.*, 13(1-2):133–170, 1980.

Terry WINOGRAD : Architectures for context. *Hum.-Comput. Interact.*, 16(2):401–419, décembre 2001. ISSN 0737-0024. URL `http://dx.doi.org/10.1207/S15327051HCI16234_18`.

Yihong XIAO, Xianzhong ZHOU et Xiaopeng HUANG : Automated semantic web service composition based on enhanced htn. *Dans The Fifth IEEE International Symposium on Service-Oriented System Engineering, SOSE 2010, June 4-5, 2010, Nanjing, China*, pages 59–63, 2010.

ChengZhi XU, Peng LIANG, Taehyung (George) WANG, Qi WANG et Phillip C.-Y. SHEU : Semantic web services annotation and composition based on er model. *Dans SUTC/UMC*, pages 413–420. IEEE Computer Society, 2010. ISBN 978-0-7695-4049-8. URL http://dblp.uni-trier.de/db/conf/sutc/sutc2010.html#XuLWWS10.

Jian YANG et Mike P. PAPAZOGLOU : Service components for managing the life-cycle of service compositions. *Inf. Syst.*, 29(2):97–125, avril 2004. ISSN 0306-4379. URL http://dx.doi.org/10.1016/S0306-4379(03)00051-6.

Stephen S. YAU et Yin YIN : Qos-based service ranking and selection for service-based systems. *Dans IEEE SCC*, pages 56–63. IEEE Computer Society, 2011.

Lei YE et Zhang BIN : Discovering web services based on functional semantics. *Dans APSCC*, pages 348–355. IEEE, 2006. ISBN 0-7695-2751-5. URL http://dblp.uni-trier.de/db/conf/apscc/apscc2006.html#YeZ06.

Tao YU, Yue ZHANG et Kwei-Jay LIN : Efficient algorithms for web services selection with end-to-end qos constraints. *ACM Trans. Web*, 1(1), mai 2007. ISSN 1559-1131.

Ehtesham ZAHOOR, Olivier PERRIN et Claude GODART : Rule-based semi automatic web services composition. *Dans 2009 IEEE Congress on Services, Part I, SERVICES I 2009, Los Angeles, CA, USA, July 6-10, 2009*, pages 805–812, 2009.

Liangzhao ZENG, Anne H. NGU, Boualem BENATALLAH, Rodion PODOROZHNY et Hui LEI : Dynamic composition and optimization of web services. *Distrib. Parallel Databases*, 24 (1-3):45–72, décembre 2008. ISSN 0926-8782. URL http://dx.doi.org/10.1007/s10619-008-7030-7.

Yajing ZHAO, Jing DONG et Tu PENG : Ontology classification for semantic-web-based software engineering. *IEEE Trans. Serv. Comput.*, 2(4):303–317, octobre 2009. ISSN 1939-1374. URL http://dx.doi.org/10.1109/TSC.2009.20.

Dmytro ZHOVTOBRYUKH : A petri net-based approach for automated goal-driven web service composition. *Simulation*, 83(1):33–63, janvier 2007. ISSN 0037-5497. URL http://dx.doi.org/10.1177/0037549707079226.

Christian ZIRPINS, Winfried LAMERSDORF et Toby BAIER : Flexible coordination of service interaction patterns. *Dans ICSOC '04 : Proceedings of the 2nd international conference on Service oriented computing*, pages 49–56, New York, NY, USA, 2004. ACM Press. ISBN 1581138717. URL http://dx.doi.org/10.1145/1035167.1035175.

Liste des figures

Liste des tableaux